UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

INSTITUT DE
CIÈNCIES
DE L'ESPAI
CSIC IEEC

FINAL DEGREE PROJECT

TFG TITLE: Validation of a Passive Bistatic Radar using Digital Satellite TV as a Source of Opportunity

DEGREE: Bachelor's Degree in Air Navigation Engineering

AUTHOR: Víctor Moreno Rodríguez

ADVISOR: Serni Ribó Vedrilla

SUPERVISOR: Jordi Berenguer i Sau

DATE: February 6, 2018

Títol: Validació d'un radar biestàtic passiu utilitzant senyals de televisió digital per satèl·lit com a fonts d'oportunitat

Autor: Víctor Moreno Rodríguez

Director: Serni Ribó Vedrilla

Supervisor: Jordi Berenguer i Sau

Data: 6 de febrer de 2018

Resum

El projecte es basa en la proposta d'un radar biestàtic passiu que utilitza els senyals de televisió digital per satèl·lit, transmesos des d'òrbites geoestacionaries, com a fonts d'oportunitat per detectar distàncies i velocitats de diferent targets. Inicialment, el principal objectiu del projecte era la detecció de pluja, però degut als pocs events de precipitació amb forta intensitat a prop de l'estació terrena del radar s'han realitzat altres experiments de detecció. El muntatge de les antenes receptores del radar, així com de tot el sistema receptor, s'ha realitzat en la teulada de l'Institut de Ciències Espacials, des d'on es tenia accés a múltiples targets com edificis, cotxes, arbres, etc.

S'ha estudiat el potencial dels senyals de televisió digital per satèl·lit i de la configuració de radar biestàtic passiu, fent emfàsi en el nostre sistema radar. L'instrument receptor existent, anomenat BIBA-SPIR (Bi-Band Software PARIS Interferometric Receiver), desenvolupat pel grup de recerca Earth Observation de l'Institut de Ciències de l'Espai, ha estat utilitzat per enregistrar y mostrejar els senyals directes i reflectits de televisió digital per satèl·lit. Tenint una antena receptora per cada enllaç, l'antena UP rebrà el senyal directe del satèl·lit i l'antena DW rebrà el senyal reflectit.

La part central del projecte es basa en el processament off-line de les senyals rebudes via software, utilitzant una llibreria de codi obert anomenada `wavpy` en C++/Fortran90 i dissenyada per a l'anàlisi i el modelatge de dades GNSS-R. S'apliquen tècniques de processat com la calibració DC, filtratge i correlacions creuades per a obtenir gràfiques de resultats com clústers i Delay-Doppler maps. La contribució principal durant aquest projecte s'ha basat en crear el codi necessari per tal de crear i definir filtres digitals i filtrar els senyals rebuts mostrejats per l'instrument BIBA-SPIR. Aquest codi ha estat inclòs en la llibreria mencionada anteriorment. També, s'ha creat una interfície gràfica en Python per proporcionar els paràmetres d'entrada i de sortida necessaris per al funcionament correcte dels programes principals utilitzats per a realitzar el processament dels senyals via software.

Els resultats experimentals obtinguts avalen la validació del radar biestàtic passiu utilitzant senyals de televisió digital per satèl·lit per detectar diferents targets i mesurar distàncies i freqüències Doppler.

Title : Validation of a Passive Bistatic Radar using Digital Satellite TV as a Source of Opportunity

Author: Víctor Moreno Rodríguez

Advisor: Serni Ribó Vedrilla

Supervisor: Jordi Berenguer i Sau

Date: February 6, 2018

Overview

The project is based on the proposal of a passive bistatic radar that uses digital satellite television signals, transmitted from geostationary orbits, as sources of opportunity to detect distances and Doppler for different targets. Initially, the main objective of the project was the detection of rain, but due to the few precipitation events with strong intensity near the radar ground station, other detection experiments have been carried out. The setup of the ground station antennas, as well as the entire receiver instrument system, was mounted on the Institute of Space Sciences roof, from where there was access to multiple targets such as buildings, cars, trees, etc.

The potential of digital satellite television signals and the configuration of passive bistatic radar has been studied, emphasizing in our radar system. The existing receiver instrument, called BIBA-SPIR (Bi-Band Software PARIS Interferometric Receiver), developed by the Earth Observation research group of the Institute of Space Sciences, has been used to record and sample direct and reflected signals of digital satellite television. Having a receiver antenna for each link, the UP antenna will receive the direct signal from the satellite and the DW antenna will receive the reflected signal.

The central part of the project is based on the off-line software processing of signals received, using an open source library called `wavpy` in C++/Fortran90 and designed for the analysis and modeling of GNSS- R. Processing techniques such as DC calibration, filtering and cross correlations are applied to obtain result graphs such as clusters and Delay-Doppler maps. The main contribution during this project was based on creating the necessary code in order to create and define digital filters and to filter the received signals sampled by the BIBA-SPIR instrument. This code has been included in the aforementioned library. Also, a graphical user interface in Python has been created to provide the input and output parameters necessary for the correct operation of the main programs used to perform signal processing via software.

The experimental results obtained support the validation of passive biestatic radar using digital satellite television signals to detect different targets and measure range and Doppler frequency.

ACKNOWLEDGEMENTS

Primero de todo me gustaría agradecer de forma especial a mi advisor, Serni Ribó, por darme la oportunidad de realizar este proyecto y por toda la ayuda desinteresada que me ha proporcionado durante estos meses. Muchas gracias por dedicar tu tiempo a mi enseñanza y por darme la oportunidad de aprender de un profesional de rigor como tú.

También, agradecer a parte de la plantilla del Institut de Ciències de l'Espai por la calurosa acogida y por la hospitalidad que me habéis dispensado. Gracias por todas las risas, consejos, historietas, anécdotas, sarcasmos y por el tiempo que hemos pasado juntos. En particular a Víctor, por sus sabios consejos y por sus masterclasses de C++ y Python; a Jaume, por sus ánimos y su ayuda en programación; a Fran, por sus consejos y por toda la ayuda que recibí para poder utilizar el receptor GPS; a Carles, por esos instantes de descanso, charlas matutinas y por sus recomendaciones durante mi estancia; y finalmente a Josep, Pepe y Carlos.

Mis agradecimientos hacia Jordi Berenguer, por sus consejos y por su disponibilidad de reunirnos cuando fuera necesario.

Gracias a mis amigos, que han sabido disculpar mis ausencias y siempre han tenido una palabra de ánimo hacia mí. Agradecer a mis compañeros de universidad, por todo lo que hemos vivido juntos.

Y por último, le doy gracias a mis padres por apoyarme en todo momento, por creer en mi, por los valores que me han inculcado y por haberme dado la oportunidad de tener una excelente educación. A mis hermanos, agradecerles sus ánimos y por hacerme sentir una referencia para ellos.

A aquellas personas que han estado presentes en mi camino y que han puesto su granito de arena para que hoy sea quién y cómo soy.

¡A todos, mi eterno agradecimiento!

"When you are inspired by some great purpose, some extraordinary project, all your thoughts break their bonds: Your mind transcends limitations, your consciousness expands in every direction, and you find yourself in a new, great and wonderful world. Dormant forces, faculties and talents become alive, and you discover yourself to be a greater person by far than you ever dreamed yourself to be."

Patañjali

A mi familia,
Antonio, Rosa, Jordi y Raquel,
con gran aprecio por su apoyo incondicional, paciencia y amor.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The capability of using illuminators of opportunity for range and Doppler target detection is of great interest for the radar community. Systems that explote the energy transmitted from these non-cooperative transmitters for radar applications are called Passive Bistatic Radar and have been successfully investigated in different configurations and using different sources of illumination. This kind of technology has several advantages for civilian and military purposes. Passive radar systems are able to operate using illuminators of opportuniy, in most cases these illuminator sources are traditional telecommunication or broadcast signals.

In particular, the alternative use of Global Navigation Satellite System (GNSS) has recently initiated a number of studies that aim to exploit this source of illumination for passive radar. Also, digital Satellite TV signals are illuminating the Earth from multiple geostationary satellites, so these signals can be used in a passive bistatic radar configuration for remote sensing of the sea-surface, as recently demonstrated by the Earth Observation research group at the Institute of Space Sciences. Initially, the main objective of the project was the detection of rain, but due to the few precipitation events with strong intensity near the radar ground station, other detection experiments have been carried out. Therefore, these were the main causes to carry out a project related to passive bistatic radars using digital satellite television signals in order to detect the range and Doppler properties for different targets.

In chapter 1 an introduction about the Institute of Space Sciences and remote sensing using signals of opportunity is provided.

Chapter 2 contains an overview about satellite digital TV signals and the selection of the opportunity transmitter for our radar system is studied. Furthermore, the potential of digital satellite television signals and the bistatic configuration of the passive radar has been studied in chapter 3.

Focusing on the Ground Station, planning and preparation of the set-up of the radar is made in chapter 4. For this task, the existing receiver instrument and developed by the Earth Observation research group will be used, called BIBA-SPIR (Bi-Band Software PARIS Interferometric Receiver). This instrument is in charge of recording the digital satellite television direct and reflected signals and to do the instrument signal processing. The perform setup of the receiving antennas will be made on the roof of the building of the Institute of Space Sciences, and consists of by the UP antenna that receives the direct signal from the transmitter satellite and the DW antenna that receives the reflected signals. Also, a study of the pointing of both antennas has been carried out, thus creating a Matlab script that provides the pointing angles of both antennas, knowing the transmitting satellite and target position.

Chapter 5 details the instrument and off-line software signal processing that is performed to obtain graphs like clusters and Delay-Doppler maps. An open source library called `wavpy` in C++/Fortran90 and designed for the analysis and modeling of GNSS-R data is used to perform the software signal processing. To determine the bistatic range of the target, it is measued the time difference (delay) of arrival of the reflected signal respect to the direct signal. The time difference between the reflected signal and the direct path signal can be calculated by cross-correlating these signals. Signal filtering techniques have been

implemented in the aforementioned software library. Also, a Graphical User Interface is designed and implemented as visual support for the software processing of the received signals.

In order to fulfill the main objective of detecting the range and doppler properties for different targets, experimental results are shown in chapter 6. Experiments have been made to detect static and dynamic targets to validate the proper work of all the passive bistatic radar system.

CHAPTER 1. PROJECT BACKGROUND

1.1. Institute of Space Sciences

The project has been developed within the Earth Observation group of the Institute of Space Sciences (IEEC - CSIC), thanks to an agreement made with the university. The research group aims to understand how the signals of opportunity can be used for remote sensing of the Earth. Its research focus is based on new remote sensing techniques in GNSS reflectometry, Radio Occultations and using digital satellite television signals.

1.2. Remote Sensing using SoOP

Nowadays, different radio frequency signals broadcast from satellites can be accessed from almost every point on the Earth. These signals can be used freely for remote sensing purposes, using techniques for inferring properties of the Earth, the enviroment or different artificial targets from measurement of signals reflectometry. The exploition of signals for a use different of its intended application and using them for remote sensing is known as remote sensing using signals of opportunity (SoOP) [14].There is a great variety of signals transmitted by satellites, such as GNSS [10], satellite digital television, among others. The use of signals transmitted from Earth, such as DVB-T can also be used as illuminator source [15] and [16].

Many contributions to the scientific-technical community about the use of GNSS signals for remote sensing of the Earth have been made during the last years. The potential use of GNSS signals of opportunity that scatter off the Earth surface for the retrieval of geophysical information is known as GNSS-R (Reflectometry), being an interesting topic for research in the last years.

In addition, there are several benefits of using digital satellite television signals transmitted by geostationary satellites for remote sensing applications. These signals can be used in a passive bistatic radar configuration for remote sensing of the sea-surface, as recently demonstrated by the Earth Observation research group at the Institute of Space Sciences [14]. Therefore, detection of rain and measuring the Doppler frequencies of different target may also be possible using digital satellite television signals.

CHAPTER 2. DIGITAL TV VIA SATELLITE

Digital Satellite Television is the broadcasting of TV signals that transmits higher quality image and sound over geostationary communication satellites. Satellite communication systems are one of the most widely used alternatives in telecommunications systems worldwide, mainly due to their wide coverage and the large bandwidth, which makes it possible to implement applications that are not possible or have important restrictions when they are implemented in other telecommunications systems. The diffusion of digital television via satellite allowed the access to the television service in remote areas, due to the lower installation cost of the receiving system and the greater coverage provided by geostationary satellites.

This chapter will present the digital satellite television signals standards (DVB-S/DVB-S2) and properties. It will also present their transmitter network structure, and all other information relevant to DVB-S/DVB-S2 as Passive Bistatic Radar illuminator.

2.1. Standards

The technical characteristics associated with the transmission and reception of digital satellite TV service depend on the standard used for that purpose. In that sense, the main standards for the mentioned service in Europe are DVB-S and DVB-S2, both still co-existing nowadays.

On the one hand, DVB-S (Digital Video Broadcasting - Satellite) was the first standard for satellite television. It uses a QPSK fixed modulation with different tools for channel coding and error correction. DVB-S is compatible with the transport frame provided by the MPEG-2 coded TV services.

On the other hand, the DVB-S2 standard it is based on DVB-S and includes improvements in spectral efficiency (bits/Hz) and adaptive as well as variable coding and modulation. Instead of constant modulation and coding of DVB-S, DVB-S2 can adopt modulation and coding depending on the weather conditions, channel interference, etc. DVB-S2 can use different modulations schemes (8PSK, 16APSK and 32APSK, being 8PSK the most used in 19.2E ASTRA constellation) that are higher than QPSK in order to increase the throughput of data.

More information about DVB-S/S2 satellite television broadcasting measurement and comparison can be found in [7].

2.2. Signal overview

In this section, the properties of digital satellite television signals that are needed for the study of their viability for passive bistatic radar application will be mentioned. The study of these properties is given of digital satellite television signal of 19.2^{o}E ASTRA and both mentioned standards.

DVB-S/DVB-S2 channels (transponders) have a bandwidth of approximately 27 MHz with center frequency between 10.7 GHz and 12.75 GHz. The signals are transmitted using horizontal and vertical polarization. In order to use the channels that are available as

efficiently as possible, signals are transmitted at both horizontal and vertical polarization. Channels at horizontal (H) polarization are transmitted in the space between vertical (V) channels and viceversa, to avoid cross-table between channels. This implies that the satellite transmits programs (channels) in horizontal and vertical polarization, which can be observed in the spectrum. As a result, there is an alternation between vertical and horizontal channel polarization.

The digital satellite television spectrum of each channel is almost flat and TV channels are transmitted in a total band that spans around 2 GHz, as can be observed in Figure 2.1.

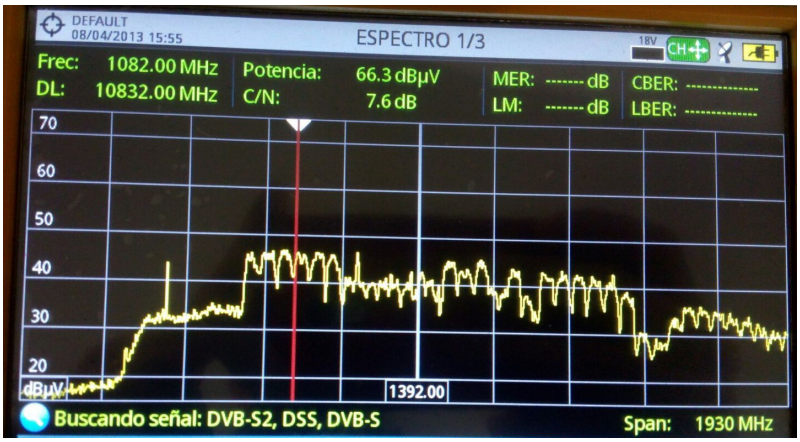


Figure 2.1: 19.2^{o}E ASTRA digital satellite television spectrum.

As will be mentioned in section 3.3., DVB-S/DVB-S2 signals have a high range resolution due to its wide bandwidth, which makes them competitives compared to other passive radar signals. Some analysis of the properties of DVB-S signal for passive radar applications is carried out in [8].

2.3. Transmitters

The satellites who are in charge of transmitting the digital satellite television signals are located at the geostationary orbit, remaining in a fixed position relative to any ground receiving location. Therefore, the receiver antenna can be pointing permanently to the the apparent fixed position of the satellite. The satellite locations are defined in longitude, since its latitude is considered null. There exist the orbital positions windows, where more than one satellite can be placed in the same longitude orbital position so that TV viewers can receive a greater choice of programs with a fixed dish antenna. For example the position 19.2^{o}E of ASTRA that is composed by a constellation of 4 satellites (ASTRA 1KR/1L/1M/1N).

The transmission of digital satellite television is based on using the satellite as a relay station, so the transmission is constituted by two communications links. The first one is the Up-link, where the television signal that needs to be distributed is first up-converted to a high microwave frequency and transmitted with large directive parabolic antennas (diameter from 9 m to 12 m) to the corresponding geostationary satellite.

In the Down-link, the satellite has changed the RF carrier frequency to Ku bands (10.7 GHz

to 12.75 GHz) of the television signal and retransmits it towards its entire coverage area. These digital satellite television signals are received by ground stations where each one has a parabolic dish (diameter from 45 cm to 120 cm) that is pointing to the geostationary satellite, a Low-Noise-Block converter (LNB) and a tuner or a decoder to watch TV.

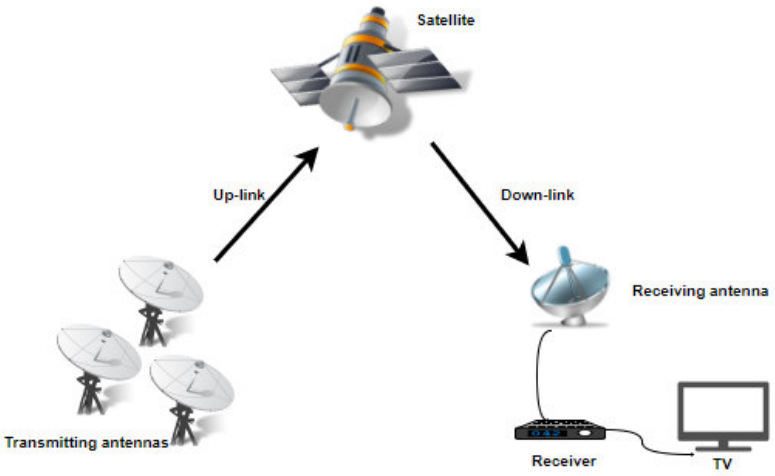


Figure 2.2: Digital Satellite Television transmission-reception links.

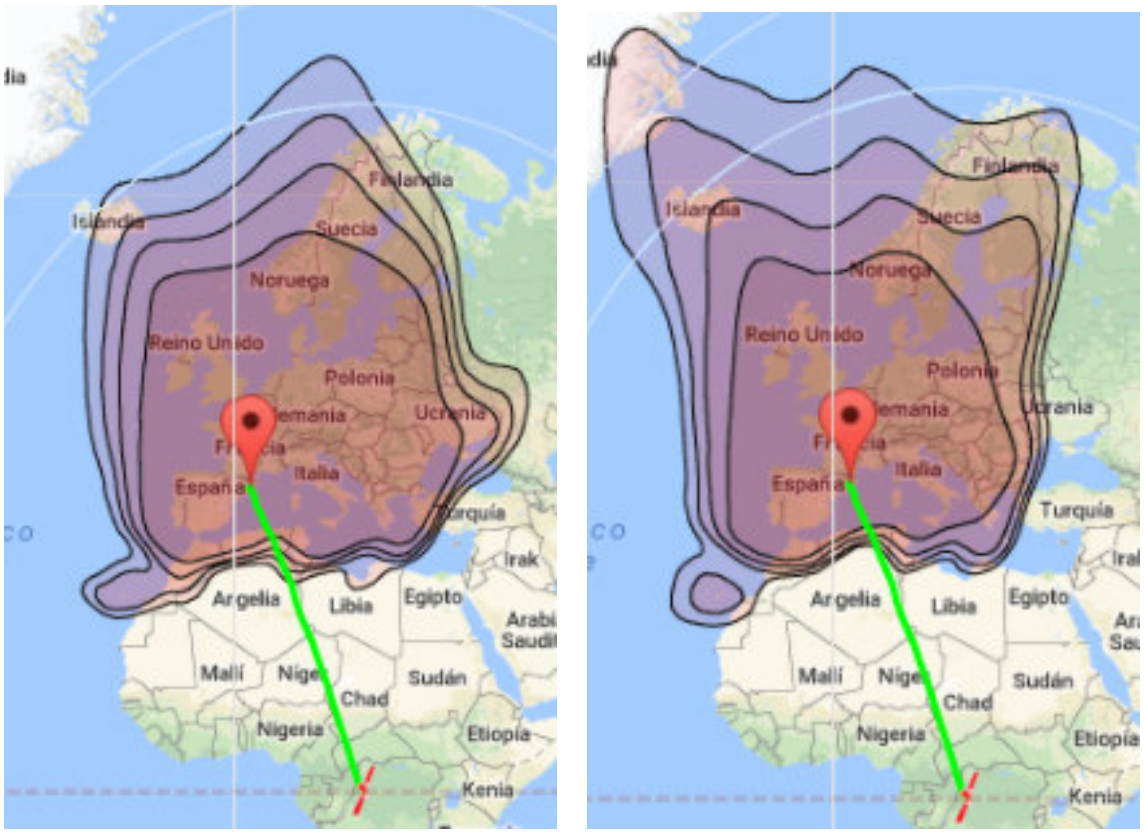
2.3.1. Choice of an opportunity transmitter

The operators of digital satellite television with the greatest presence in the region where our antennas have been installed are Hispasat and ASTRA. Therefore, it is necessary to choose between them which is the best constellation of satellites that provide us with the best direct path signal-to-noise ratio and the best pointing antenna conditions.

The 30^{o}W Hispasat satellite constellation provides the highest EIRP, but the pointing antenna conditions is worth than pointing to the 19.2^{o}E ASTRA constellation. Taking into account that the satellite signal is only received if there is line of sight (LOS), the obstacles that can have block the line of sight will be studied in each transmitter case. A precise pointing of the directive UP antenna to different constellations has been done, monitoring the RF power of the received signals with a spectrum analyzer. Hence, it has been decided to choose the 19.2^{o}E ASTRA satellites constellation as transmitters of opportunity.

As will be discussed later, the fact of choosing this opportunity transmitter allows us to receive signals from digital satellite television with carrier-to-noise ratio above the established minimum. Also, it permits us to choose channels with different DVB-S and DVB-S2 standards (using the predefined filters of L1 and L5 of the receiving instrument) and from different transmitters (1KR and 1M). The received digital satellite television signals are transmitted by the 1KR and 1M satellites from the 19.2^{o}E ASTRA constellation.

As following, the coverage of 19.2^{o}E ASTRA 1KR and 1M satellites can be observed, where the transmitted EIRP in the area that covers the Institute of Space Sciences (Cerdanyola del Vallès) is 50 dBW.



(a) 19.2°E ASTRA 1M.

(b) 19.2°E ASTRA 1KR.

Figure 2.3: 19.2°E ASTRA 1KR and 1M coverage, from https://www.satbeams.com/footprints.

CHAPTER 3. PASSIVE BISTATIC RADAR

The IEEE Standard 686-2008 [1] defines bistatic radar as a radio detecting ranging system that uses electromagnetic waves to detect and track targets, *using antennas for transmission and reception at sufficiently different locations that the angles or ranges from those locations to the target are significantly different*.

Unlike classical bistatic radars, Passive Bistatic Radars (PBR) make use of transmitters of opportunity instead of a cooperative transmitter, so they are intrinsically a non-cooperative system. The target detection and the estimation of its parameters is based on a comparison between received signals, direct signal of the transmitter of opportunity and the scattered signals.

Due to the advantage of not needing a transmitter specifically designed for PBRs and they result in cheaper systems, they have a significant attraction for the scientific-technical community and many advances are being made on these systems. Table 3.1 shows the main advantages and disadvantages of the PBR systems:

Table 3.1: Advantages and Disadvantages of Passive Bistatic Radars

Advantages	**Disadvantages**
Relatively low cost	Inherent inability of the design on the transmitter
Low vulnerability	Complicated geometry compared with monostatic radar
No demand for frequency allocation	Advanced hardware and software technology
Military applications:	Some transmitter's ambiguity functions
stealth and immune to Anti-Radiation Missiles	are not proper for radar operation
No jamming Reduced impact on the environment	
Large number of transmitters	
Covert operation	
High transmitters coverage	

There are several types of illuminators of opportunity, such as Global Navigation Satellite Systems (GNSS), Terrestrial and Satellite Digital Television, FM radio, etc. The reason for using digital satellite TV signals is that the available power, bandwidth, number of transmitters and coverage is greater than in GNSS. In this chapter we will describe the basic elements of a radar which uses the transmissions of Satellite Digital Television already present in the region of interest as the radar waveform.

3.1. Geometry definition

The bistatic geometry of the PBR system is based on the separation in distance of the transmitter and receiver. This geometry affects the performance of the system in terms of detection range and resolution.

The opportunity signals are unknown by the radar receiver, so the PBRs usually need two

receiving antennas. The reference receiver antenna, which is steered toward the transmitter of opportunity, collects the reference signal and the surveillance receiver antenna collects the signals backscattered from the pointing area. The receiving system may also consist of a single receiving antenna, with the disadvantage that the data collector system of the receiver will be more complex.

The coordinate system used to describe the geometry will be in a two dimensional situation with a north referenced coordinate system. It must be said that in the experiments carried out for this project, two receiving antennas have been used, but both are considered to be in the same geographical position due to its proximity. An illustration of a passive bistatic radar can be seen in Figure 3.1.

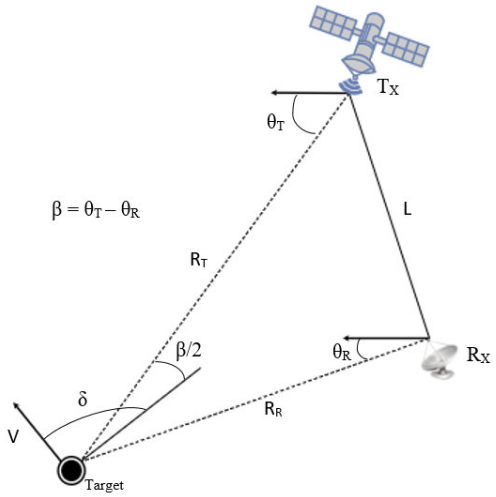


Figure 3.1: Bistatic radar geometry (also applied for PBR).

T_x and R_x are the position of transmitter and receiver, respectively, being separated with a baseline direct path line L. The distance R_T is the range from the transmitter to the target, while R_R is the range from the target to the receiver. The sum of both mentioned distances will be the distance travelled by the opportunity signal scattered on the target.

The bistatic angle (β) is one of the important parameters that characterizes the bistatic radar and affects to the system performance. It can vary between 180^o when the target is on the baseline to 0^o when the target is on the pseudomonostatic region, a special bistatic geometry that approximates monostatic operation or in practice it defines bistatic operation near the extended baseline. Also the transmitter and receiver looking angles, can be defined as θ_R and θ_T respectively. In addition, the target's velocity vector projected onto the plane has magnitude V and direction with aspect angle δ.

When omnidirectional antennas are used to receive the scattered signals, when $R_T + R_R$ is measured with bistatic configuration, the target position can be located on every point of a constant range ellipse with focal points in the transmitter and receiver. In the case studied in this project, the receiving antenna of the signals reflected in the targets is directive, so there is no uncertainty of the location of the targets.

3.2. Radar equation

The bistatic radar equation can be used to predict the performance of the passive bistatic radars. Indeed, the quantity of interest is the power received (sensitivity) at the receiver antenna for a given geometry, target and transmitter. It is known, that the power of transmitted signal is expressed using the **E**ffective **I**sotropic **R**adiated **Power** ($EIRP = P_T \eta D_T$). The satellite transmitter transmits a signal of bandwidth B at wavelength λ, with a peak transmit power P_T and with an antenna of directivity D_T and efficiency η. The target is located at a distance R_T from the transmitter and a distance R_R from the receiver has a bistatic radar cross section σ. The bistatic radar cross section, also expressed as BRCS, quantifies how detectable a target is in a bistatic configuration, expressing indirectly its equivalent size and ability to reflect the transmitted signal in the direction of the receiver compared to an isotropic reflector. The BRCS has units of area.

It is considered that the scattered signal spreads as a spherical wave and that it is intercepted by the receiver antenna of directivity D_R and efficiency η_R. Being in a non-ideal scenario, where non free-space is assumed, we can consider a loss factor $F_L > 1$. This factor takes into account effects such as attenuation losses, interferences, diffraction, multipath and other environmental factors. Concerning to the receiver system losses, a loss factor is applied $L_s > 1$. As a result, the radar equation relationship can be developed as in equation (3.1).

$$P_R = \frac{EIRP}{4\pi R_T^2}\sigma\frac{\eta_R D_R}{4\pi R_R^2 L_s F_L}\frac{\lambda^2}{4\pi} \tag{3.1}$$

The noise power at the receiver is given by 3.2.

$$P_N = k_B T_0 B F \tag{3.2}$$

Where F is the receiver noise factor, B is the receiver bandwidth, T_0 = 290 K is the reference room temperature and k_B = 1.38 x 10^{-23} W/(Hz K) is the Boltzmann's constant.

SNR is defined as the measure of signal strength relative to the background noise. So the SNR is computed in (3.3) as the ratio between the power received and the thermal noise power (P_N) at the receiver.

$$SNR = \frac{P_R}{P_N} = \frac{EIRP\lambda^2\eta_R D_R\sigma}{(4\pi)^3 R_T^2 R_R^2 L_s F_L}\frac{1}{k_B T_0 B F} \tag{3.3}$$

3.2.1. Pulse compression

To increase the probability of detection for a given scenario it is necessary to maximise the SNR of the target returns. So, to increase the SNR a possible solution is to integrate over time. It can only be applied if the target and receiver are stationary, in this scene the target samples add in phase.

Taking into account argumentations from [2], B from (3.3) is replaced by the inverse of the coherent processing interval defined as T_{inc}.

$$B = \frac{1}{T_{inc}} \tag{3.4}$$

Increasing the reflected power in the target and incoherently increasing the noise power, the resulting is an increase in the target reflected signal-to-noise ratio by a factor of $10\log_{10}(T_{inc})$ in dB. The pulse compression is equivalent to a matched filter processing.

3.3. Range relationship

The resolution of a radar is defined as the minimum separation between point targets that allows them to be individually distinguished by radar signal processing in both range and Doppler. The bistatic range is defined as the sum of the transmitter-target range (R_T) and target-receiver (R_R). As has been said in Section 3.1., with a bistatic geometry the target can be located on an iso-range ellipse with focal points in the transmitter and receiver (in the case of using omnidirectional antennae). This means that the range resolution has dependence on the bistatic geometry.

Because of the bistatic radar configuration, PBR has range cells that are given by the difference of the range resolution in two concentric ellipses (Figure 3.2), so the range resolution becomes dependent of the bistatic angle (β). The range resolution of all radar systems is given by the bandwidth of the radar signal, or the opportunity signal bandwidth in case of a PBR.

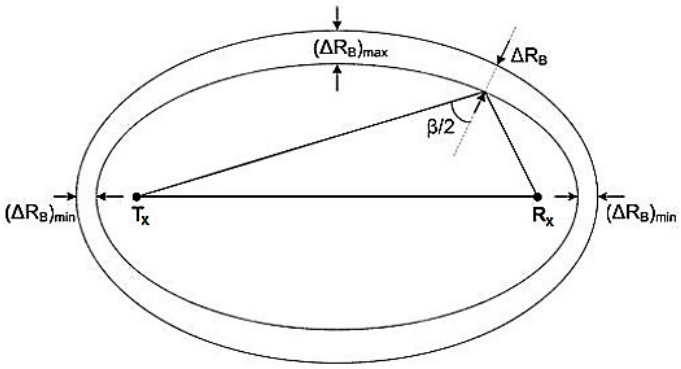


Figure 3.2: Bistatic radar range resolution with an iso-range ellipse contour.

Furthermore, the bistatic range resolution, ΔR_B, can be approximated as in equation (3.5). It can be observed that the best case of range resolution occurs when β is null and the range resolution becomes minimum, with a pseudomonostatic scenario (when T_x, R_x and the target are approximately located in the baseline) and non a bistatic one. The worst case of bistatic range resolution is when it becomes maximum and that occurs when the bistatic angle approaches to 180°.

$$\Delta R_B = \frac{c}{2Bcos(\beta/2)} = \frac{\Delta R_M}{cos(\beta/2)} \tag{3.5}$$

Where c = 299,792,458 m/s is the speed of light, B is the processed signal bandwidth and β the bistatic angle of the configuration. ΔR_M is defined as the range relationship in the

monostatic case (transmitter and receiver are located in the same place). As can be observed, the range resolution depends on the bandwidth and on the bistatic configuration. With the same bistatic configuration and knowing that the bandwidth of the measured digital satellite TV signal as mentioned above is larger than other illuminators of opportunity. The great advantage of digital satellite TV signals large available power and broader total bandwidth, compared with other illuminators of opportunity.

A specific calculation of the range resolution will be made for digital satellite television signals and will be compared with digital terrestrial television signals (DVB-T) and a type of GNSS signals.

The bandwidth of a DVB-S/S2 transponder can take different values, in these example a frequent value of 27 MHz is chosen. After the instrument and software processing of the signals, in our case the bandwidth of the processed signal is 20 MHz (digital filter of 10 MHz cutoff frequency in baseband is applied). In our Ground Station location, the 19.2°E ASTRA satellites have an elevation of approximately 39°, so the minimum bistatic angle is β = 39° (when GS and target are at the same altitude). The resulting minimum range resolution with the explained configuration is ΔR_B = 15.91 m.

For the same bistatic angle, a DVB-T passive bistatic radar capable of selectively receive a single UHF band TV-signal channel of 8 MHz bandwidth has a range resolution of ΔR_B = 39.75 m. In the case of one type of GNSS at L1 (CA code), for the same bistatic angle and considering a pulse width of $2\mu s$ after correlating, the range resolution is ΔR_B = 318 m.

3.4. Velocity and Doppler shift

The Doppler shift is the rate of change of the total path length of the reflected signal, normalized by the wavelength. For a bistatic radar, the Doppler shift of a reflected signal without taking into account the relativistic effects is given by (3.6).

$$f_B = \frac{1}{\lambda}\left[\frac{d}{dt}(R_T + R_R)\right] \tag{3.6}$$

In reference to Figure 3.1, it is know that the target's velocity has magnitude V and aspect angle δ. In the case of static transmitter and receiver, the projections of the target velocity vector will be onto the transmitter-target line of sight and onto the target-receiver line of sight.

$$\frac{dR_T}{dt} = V cos(\delta - \beta/2) \tag{3.7}$$

$$\frac{dR_T}{dt} = V cos(\delta + \beta/2) \tag{3.8}$$

After combining equations (3.7) and (3.8), the bistatic Doppler shift for static transmitter and receiver is described in equation (3.9).

$$f_B = \frac{2V}{\lambda} cos(\delta) cos(\beta/2) \tag{3.9}$$

Also, the bistatic velocity or the projected target velocity can be defined as follows:

$$v_B = V cos(\delta) cos(\beta/2) \tag{3.10}$$

$$f_B = \frac{2v_B}{\lambda} \tag{3.11}$$

In Figure 3.3 it is shown the normalized Doppler shift $f_b(\cos\delta, \cos\frac{\beta}{2})$ as a function of the aspect angle of the target velocity vector for different bistatic angles. Can be observed that monostatic Doppler shift occurs in the case of β = 0º and the bistatic Doppler shift is nullified when β = 180º.

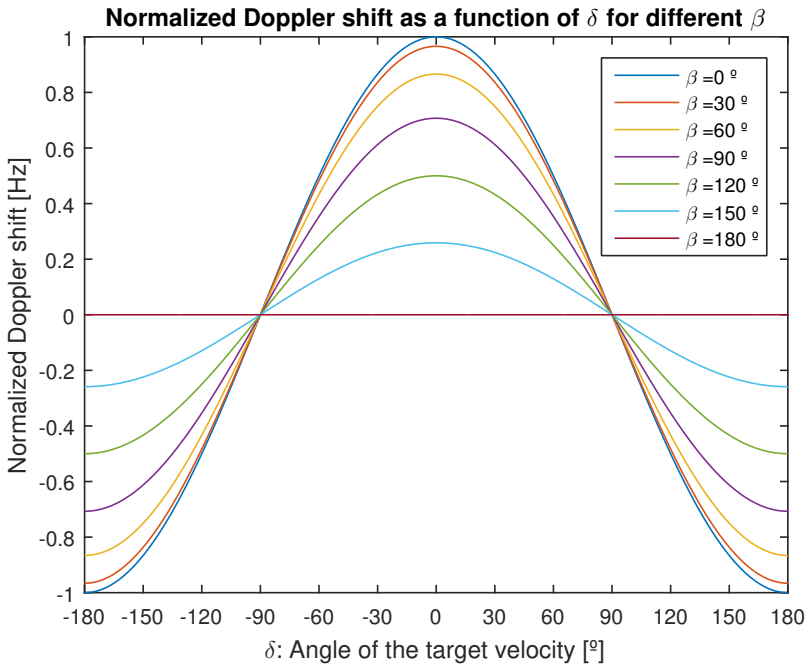


Figure 3.3: Normalized Doppler shift as a function of δ for different β

3.5. Ambiguity function

The signal processing of the proposed radars consists in cross-correlating the direct and reflected signals. This is equivalent to the matched filter processing when the SNR of the direct signal is sufficiently large [9]. The investigation of the correlation properties of the used illuminator signal is essential. In order to analyze the potential opportunities of the used illuminator signals, the ambiguity function and resultant ambiguity surface plot provide a useful tool for analysis.

The ambiguity function depends largely on the radar waveform (and in case of PBR systems, on the signal of opportunity waveform), being the ability of a radar to detect targets amongst returns from other objects (clutter) in a region of interest and the ability to determine parameters of these targets, such as range, bearing, size and velocity.

In general, the ambiguity function for a transmitted signal $x(t)$ is a 2-D autocorrelation given by the following (3.12):

$$R_{x,x}(\tau, f_D) = \int_{-\infty}^{+\infty} x(t)x^*(t-\tau)e^{-j2\pi f_D t}\mathrm{d}t \tag{3.12}$$

Where τ and f_D are the delay and doppler frequency, respectively. $x(t)$ is the transmitted complex signal, also known as reference signal, and $x^*(t)$ is its complex conjugate. Some properties of the ambiguity function include:

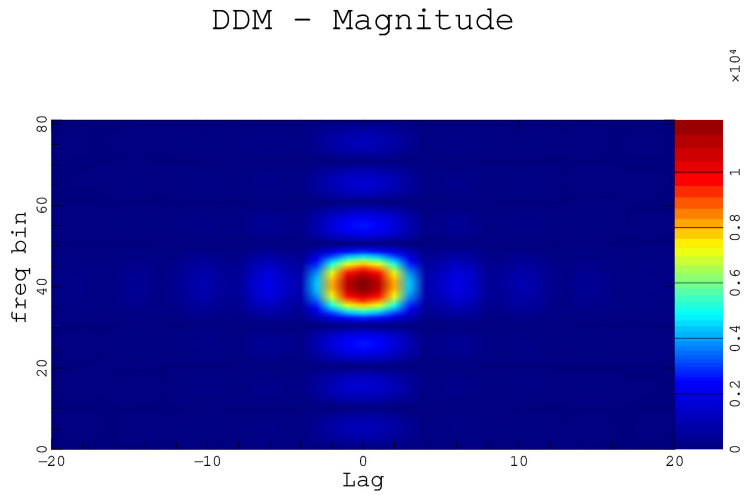
- The maximum ambiguity function value occurs at the origin, since the received signal arrives with the same delay and zero Doppler shift at the receiver. Also, the maximum ambiguity function value equals to the total power of the signal (in our case it is used a digital filter that applies a unitary gain in order to not magnify the magnitude): $|R_{x,x}(\tau, f_D)|_{max} = R_{x,x}(0,0) = P_x$.
- The ambiguity surface is symmetric along both time delay and frequency axes: $R_{x,x}(\tau, f_D) = R_{x,x}(-\tau, f_D)$ and $R_{x,x}(\tau, f_D) = R_{x,x}(\tau, -f_D)$.

When the spectral power density has a rectangular shape, the cross-correlation function has a sinc shape along the τ domain.

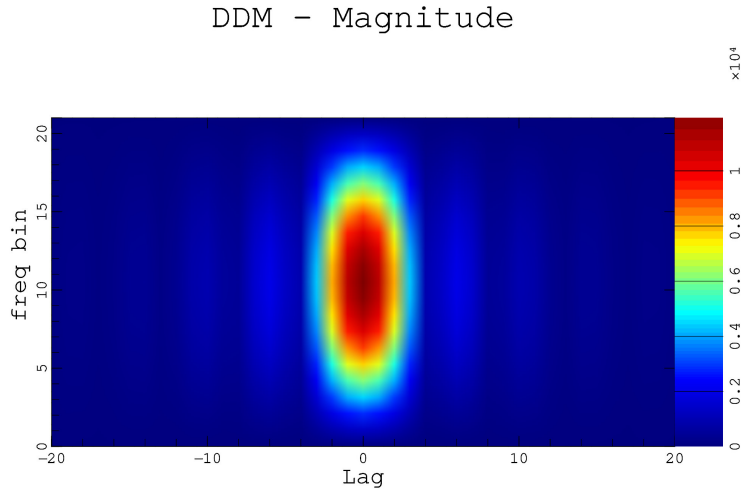
3.5.1. Measured ambiguity function

In order to verify that digital satellite television signals are a good source of opportunity for use in passive bistatic radar systems, it has been decided to measure the ambiguity function for an specific case. Using only the UP antenna pointing to the 19.2°E ASTRA satellites (pointing angles of 155.11° azimuth and 38.97° elevation), the auto-correlation of the direct path received signal is computed by taking the channel obtained with L1 filter of the instrument. For the calculation of the ambiguity function the techniques of software signal processing have been used, which will be mentioned later in 5.2..

Figure 3.5.1. shows the measured ambiguity function (in Delay Doppler Map form) of the digital satellite TV signal transmitted by 1KR 19.2°E ASTRA satellite. This corresponds to the TV channel with central frequency of 11318 MHz. The lags are defined as the temporal distance between both correlated signals (distance delay), it will be explained in more detail later.



(a) Ambiguity Function from -40 Hz (freq bin 0) to 40 Hz (freq bin 81).



(b) Ambiguity Function zoomed in frequency, from -10 Hz (freq bin 0) to 10 Hz (freq bin 21).

Figure 3.4: Ambiguity function of the DVB-S/DVB-S2 signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

Delay Doppler Maps (DDM) are computed using the `wavpy` software library [13]. The "freq bin" axis is always defined between opposed frequencies, being $0Hz = \frac{f_{max}-f_{min}}{2}$ freq bin the central frequency. Therefore, the defined axis will start at 0 freq bin and it will end at the $(f_{max} - f_{min} + 1)$ freq bin.

It's been said said that the auto-correlation calculation acts as if the signal has passed through a matched filter. Taking into account this principle, the ambiguity function represented in a Delay-Doppler map represents the output of a matched filter bank where it is composed of the reference signal and its Doppler shifted copies.

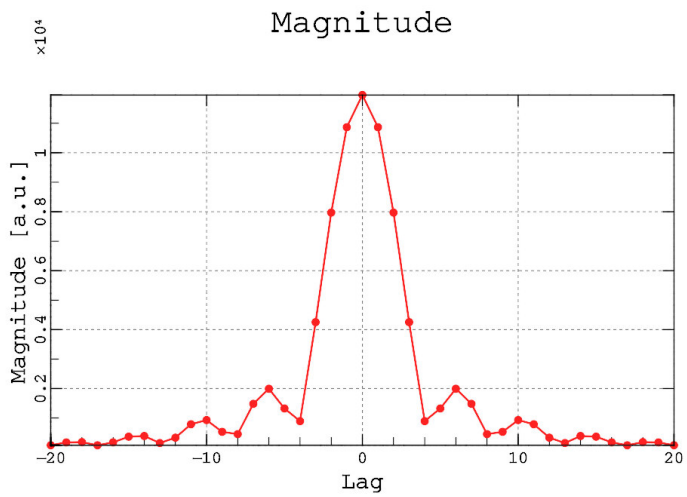


Figure 3.5: Lag (time delay) profile of the figure 3.4(a) and ambiguity function at 0 Hz Doppler frequency.

It can also be observed that the used illuminator signal has a good correlation properties and Doppler shifted copy of itself, only having a peak in both zero time and doppler frequency. In general, it can be said that a signal has advantageous correlation properties if the structure of the signal is noiselike with broadband smooth spectrum, as the digital satellite television signals. According to the ambiguity function shown before, the digital satellite TV provides appropiate signals for being used in a passive bistatic radar. There is a dynamic range enough to identify correlation peaks provoked due to some target reflections.

3.6. Direct Signal Interference

A challenge with the passive radars is to apply algorithms in order to suppress the strong direct signal incoming at the surveillance antenna (pointing to the scatterers), emitted from the transmitter, in order to be able to detect targets with weaker reflected signals. The direct signal at zero Doppler emitted from the transmitter does also leak into the secondary lobes of the antenna that points to the scatterers. This may hide weaker echoes that are below the secondary lobes of the ambiguity function.

A potential countermeasure capable of neutralizing the direct signal interference is to place a big metal shield in the direction of the transmitter. This is not possible, and there may be contributions of the direct signal from other incoming angles, caused by reflections from, e.g., nearby buildings and terrain.

A more practical solution might be to use direct signal interference mitigation techniques such as [22], [23], [24], [25] and [26]. Finally, a DSI mitigation technique of the `wavpy` library [13] has been used in some experimental results.

CHAPTER 4. INSTRUMENT AND SETUP

4.1. Receiver instrument

The existing receiver instrument, called BIBA-SPIR (Bi-Band Software PARIS Interferometric Receiver) [12], developed by the Earth Observation research group of the Institute of Space Sciences has been used to record digital satellite television direct and reflected signals. One antenna has been used for each link. The UP antenna receives the direct signal from the transmitter satellite and the DW antenna receives the reflected signal.

This instrument consists of an RF receiver with two dual-band channels (UP,DW), A/D converters and programmable local oscillator frequency and bandwidth. BIBA-SPIR is capable to manage a sustained recording data rate of 320 MB/s, with a highly accurate GPS time reference. For that a GPS antenna is used to obtain the time and positioning information.

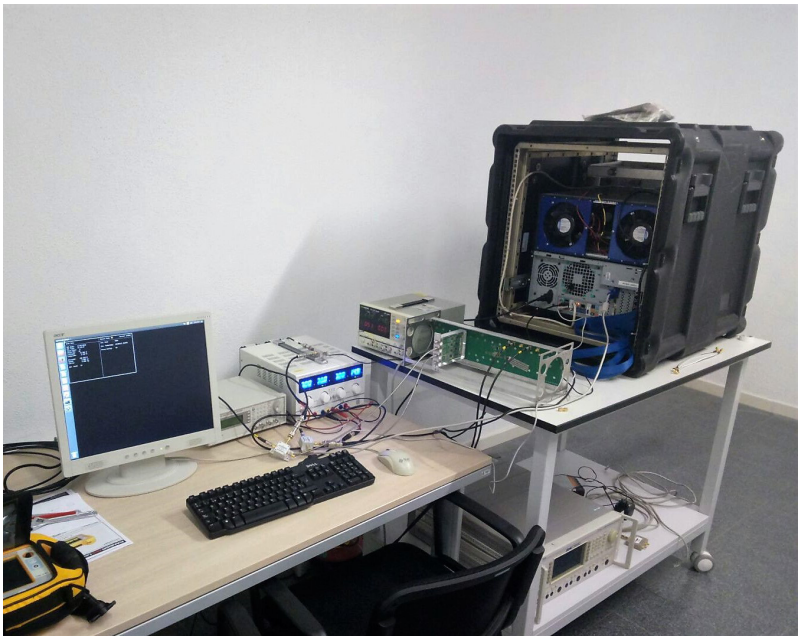


Figure 4.1: BIBA-SPIR instrument setup.

The sampled data of the recorded signals is stored into files of 1 GPS second, where the in-phase and quadrature components of the signals are digitized with one bit per sample and are sampled at a rate of 80 Msamples/s. These generated files will be used to make the software signal post-processing in section 5.2.

In Figure 4.1, it can be observed the BIBA-SPIR instrument with its corresponding RF front end, power supplies, control computer and the necessary wiring to feed the LNBs, the GPS antenna and to receive the direct and reflected digital satellite TV signals. In Figure 4.2 it is shown a more detailed view of the BIBA-SPIR instrument dual band front-end (simultaneous L1 and L5 frequencies).

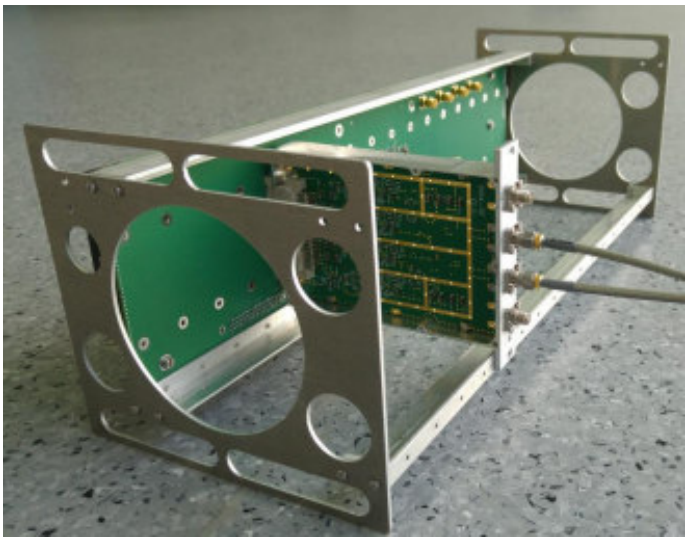


Figure 4.2: BIBA-SPIR front-end (image extracted from [12]).

Also a Graphical Interface User (GUI) is used to start and stop the recording, where L1-UP/DW and L5-UP/DW parameters are configurable.

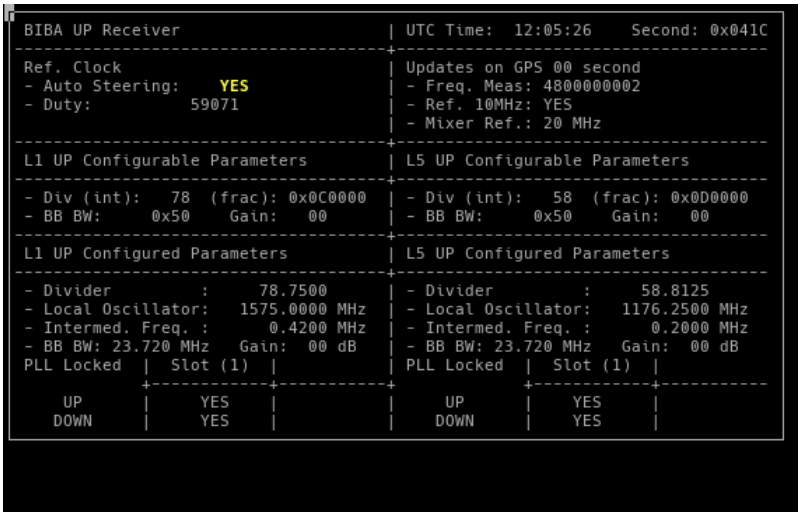


Figure 4.3: SPIR's GUI with configurable parameters.

4.2. Ground Station setup

For the accomplishment of these experiments a Ground Station (GS) has been set-up, where the assembly of the receiving antennas has been made on the roof of the Institute of Space Sciences (ICE-CSIC) building. This location provides the opportunity of easily pointing the uplink antenna (UP) that receives the ASTRA 19.2E satellite digital TV signals and the downlink antenna (DW) that receives the reflected signals. The DW antenna can be pointed into many possible scatterers as buildings, trees, cars, even planes/helicopters and rain clusters.

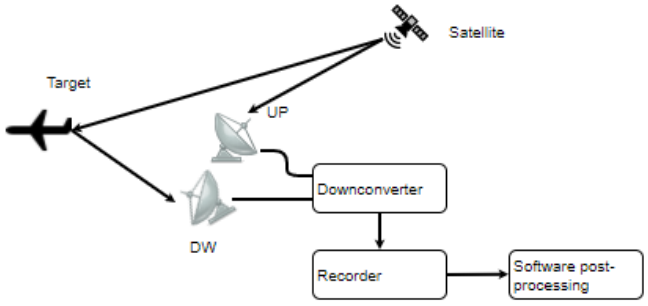


Figure 4.4: Explanatory diagram of the PBR setup.

As can be observed in Figure 4.2., a commercial parabolic dish antenna of 60 cm diameter is used to collect the direct path signal (UP) provided by the satellite and in the focus of the dish is located an LNB (SMW PLL-LNB ext. 10 MHz ref B) in charge of amplifying the signal received from the satellite, shift it to L-band and distribute it, through coaxial cable, to the BIBA-SPIR instrument.

(a) DW antenna with broader beam.

(b) Directive DW antenna.

Figure 4.5: Ground Station UP and DW antennas.

An identical LNB is used for the downlink antenna (DW), employed to collect the reflected path signals. The difference with respect to the UP antenna is that in the tests realized the DW antenna was only the LNB feedhorn (without dish), and was used first to have a broader beamwidth , and later the parabolic dish was added to have more directivity and be able to have more precision pointing to the scatterers. Beamwidth of the feedhorn is

estimated to be approximately 30^o and the beamwidth of the used parabolic dish antennas is approximately 4.5^o.

The initial purpose of mounting this set-up on the roof of the institute building was to collect data in rain scenarios and try to detect rain clusters. Due to the fact that during the realization of this project there have not been events of high intensity precipitation with clouds close to our setup, signals reflected in other targets have been collected as can be seen in the experimental results in chapter 6.

4.2.1. Pointing of the UP antenna

The UP antenna has been pointed to the 19.2^{o}E ASTRA satellites constellation, in order to receive satellite digital TV signals. For this purpose, it is necessary to find the satellite/s that provide the best coverage according to the geographical location of the Ground Station and compute the angles of azimuth and elevation required for pointing the antenna to the satellite.

Digital TV satellites are defined as geostationary with a non retrograde circular orbit in the equatorial plane (null inclination and eccentricity). These satellites theoretically remain motionless on a given point on Earth, although they have a window of motion, so they have the same angular velocity as Earth. The ground station antenna uplink (UP) antenna does not need adjusting, because no tracking is necessary. Therefore, the following characteristics of geostationary satellites can be defined:

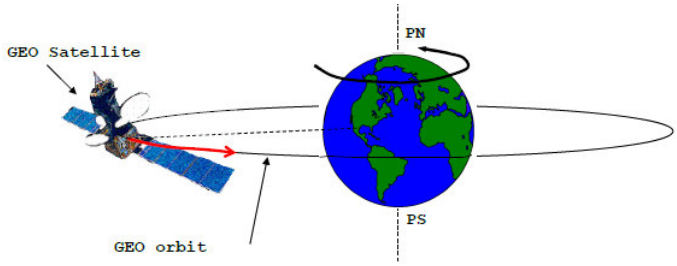


Figure 4.6: Example of geostationary orbit.

Taking into account that a sidereal day is the period of the Earth rotation over its axis and its value is T = 23h 56min 4.1s = 86164.1s. In a perfect Keplerian movement with geocentric gravitational constant μ = 398600.4418 $\frac{km^3}{s^2}$, the nominal orbit radius r can be computed with the following equation:

$$T = 2\pi\sqrt{\frac{r^3}{\mu}} \tag{4.1}$$

Moreover, can be computed the satellite GEO orbital height (in this case using nominal Earth radius R_T, although in the made scripts has been computed a specific Earth radius for each Earth location).

$$r_0 = r - R_T = 35786.157km$$

The digital TV satallites that provide the best coverage in the Ground Station region are from that ones from ASTRA are situated in an orbital location of 19.2^{o}E, as the 1M/1KR. In

order to compute the pointing angles (azimuth and elevation) of the uplink antenna, must be calculated the distance between the Ground Station and the chosen satellite using the Earth-Centered-Earth-Fixed (ECEF) reference system.

The location of the Ground Station is obtained from a GPS receiver, so it was defined in geographical coordinates: Latitude, Longitude and Altitude. The location of the satellite it is also in the same reference system, but for simplifying the calculations a reference system change is made to ECEF.

Using **W**orld **G**eodetic **S**ystem **84** Earth model with semimajor axis $a = 6378137m$ and eccentricity $e = 0.0818$, the prime vertical radius of curvature (N) and the ECEF coordinates (x, y, z) can be defined.

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2(\phi)}} \tag{4.2}$$

$$x = (N + h)\cos(\phi)\cos(\lambda) \tag{4.3}$$

$$y = (N + h)\cos(\phi)\sin(\lambda) \tag{4.4}$$

$$z = ((1 - e^2)N + h)\sin(\phi) \tag{4.5}$$

Where ϕ, λ and h are the latitude, longitude and altitude, respectively.

Table 4.1: Geographical locations.

	Latitude [º]	**Longitude [º]**	**Altitude [m]**
Ground Station	41.500436	2.110422	138
ASTRA 1M/KR	0.000000	19.200000	35,786,000

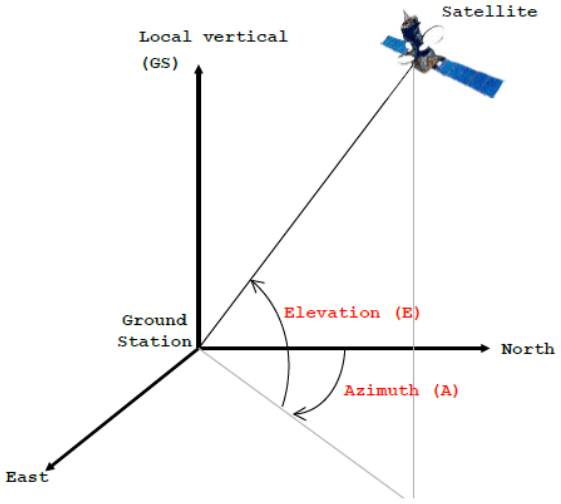


Figure 4.7: Pointing Angles of the Uplink antenna

Knowing the value of the latitude of the Ground Station (ϕ_{GS}) and the relative satellite longitude (λ_{sat}), the pointing angles azimuth and elevation can be computed.

$$\alpha = tg^{-1}\left(\frac{\tan(\lambda_{sat})}{\sin(\phi_{GS})}\right) \tag{4.6}$$

The azimuth (α) is the angle between the ground station meridian and the local vertical plane that contains the satellite pointing direction. It is measured from the North and ranges from 0º to 360º. The azimuth angle computation varies depending on the geographical situation of the Ground Station:

North - West: A = 180º - α
North - East: A = 180º + α
South - West: A = α
South - East: A = 360º - α

The elevation (ε) is the angle between the satellite pointing direction and the local horizon of the Ground Station.

$$\varepsilon = \frac{\cos(\lambda_{sat})\cos(\phi_{GS}) - \frac{R_T}{R_T+R_0}}{\sqrt{1-\cos^2(\lambda_{sat})\cos^2(\phi_{GS})}} \tag{4.7}$$

Table 4.2: UP antenna pointing angles and distance to satellites.

GS - 1M distance [m]	UP true azimuth [º]	UP true elevation [º]
37,851,814.71	155.11	38.97

4.2.2. Pointing of the DW antenna

The directive DW antenna must be used when it is necessary to perform a precise pointing towards the target. To calculate the pointing angles of the DW antenna, the first step is to know the geographical coordinates (ϕ,λ and h) of the Ground Station and of each target.

The purpose is to convert the direction vector ($\bar{\rho} = \bar{r} - \bar{R}$ vector between Ground Station and target) in ECEF coordinates to Azimuth and Elevation given geocentric latitude, east longitude, and height above the ellipsoid of the Ground Station.

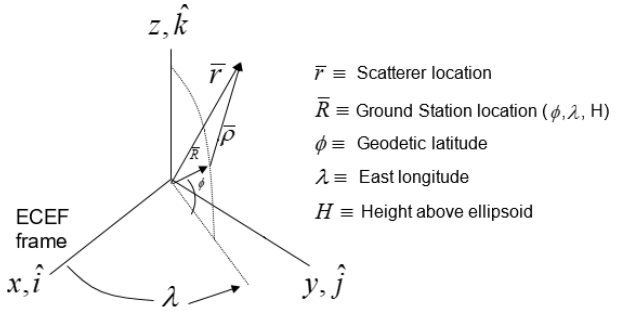


Figure 4.8: ECEF reference system with direction vector, based on images of [6].

A reference system must be created at the point where the pointing is performed, so the reference system (Antenna-Centered-Antenna-Fixed) is created with origin in the Ground Station. In the ACAF reference system there are three unitary axis: $\hat{S}$ (pointing to the South) , $\hat{E}$ (pointing to the East), $\hat{S}$ (pointing to the Zenith).

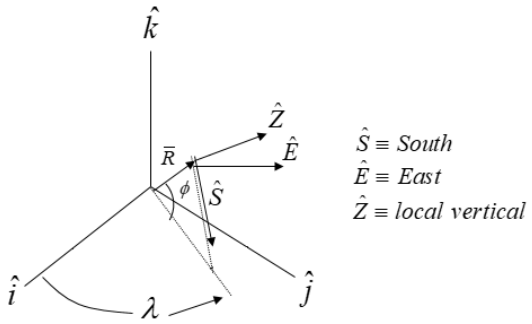


Figure 4.9: ACAF reference system, based on images of [6].

A rotation matrix is calculated in order to make base changes between the ECEF and ACAF reference systems.

$$\begin{bmatrix} \hat{S} \\ \dot{E} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} \sin(\phi_{GS})cos(\lambda_{GS}) & \sin(\phi_{GS})sin(\lambda_{GS}) & -\cos(\phi_{GS}) \\ -\sin(\lambda_{GS}) & \cos(\lambda_{GS}) & 0 \\ \cos(\phi_{GS})\cos(\lambda_{GS}) & \cos(\phi_{GS})cos(\lambda_{GS}) & \sin(\phi_{GS}) \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} \tag{4.8}$$

In order to get S, E and Z, the ECEF coordinates (x, y, z) of the direction vector $\bar{\rho}$ have been replaced in the transposed vector $[\hat{i}, \hat{j}, \hat{k}]$. Therefore, the direction vector is transformed to the ACAF reference system: $\bar{\rho} = S\hat{S} + E\hat{E} + Z\hat{Z}$.

$$S = -\rho\cos(E)\cos(A) \tag{4.9}$$

$$E = \rho\cos(E)\sin(A) \tag{4.10}$$

$$Z = \rho\sin(E) \tag{4.11}$$

The elevation (ε) and azimuth (α) can be computed. The function ***atan2(y,x)*** is the arctangent function with two arguments, giving the angle between the positive x-axis of a plane and the point given by the coordinates (x, y) on it. This function returns the appropriate quadrant of the computed angle.

$$\varepsilon = \sin^{-1}\left(\frac{Z}{\rho}\right), -90^{o} \leq \varepsilon \leq 90^{o} \tag{4.12}$$

$$\alpha = atan2(E, -S), 0 \leq \alpha \leq 360^{o} \tag{4.13}$$

CHAPTER 5. SIGNAL PROCESSING

5.1. Instrument processing

In this section, the signal processing performed by the instrument will be explained briefly. It is known that the frequency band of digital satellite television is between 10.7 and 12.75 GHz approximately. So, the LNB used B is in charge of performing a frequency band conversion with a local oscillator whose value has been previously chosen for the low band (f_{LO_X} = 9.75 GHz) so that the signals can be transmitted by the coaxial cables. Then, the BIBA-SPIR proceeds to filter the received signals, with a 40 MHz filter at L-band, and shifts the signals to baseband (applying a local oscillator programming frequency equals to $f_{LO_{L_1}}$ = 1575.0 MHz or $f_{LO_{L_5}}$ = 1176.25 MHz). Finally, the signals are I/Q (In-phase/Quadrature) sampled with 4 bits/sample and a sampling rate of 80 Msamples/second. In Figure 5.1 it can be observed the block diagram of the BIBA-SPIR and in Figure 5.2 a graphic explanation of the instrument processing is shown.

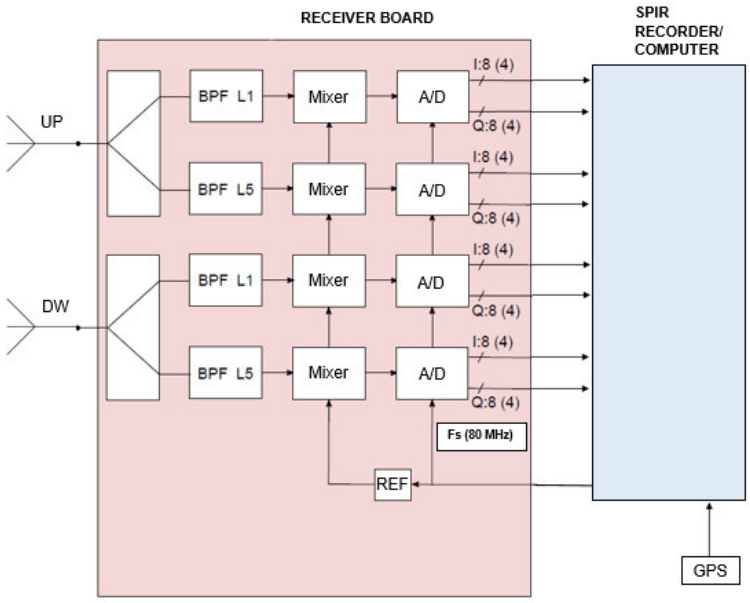


Figure 5.1: Dual-band SPIR (BIBA-SPIR) block diagram, having [12] as reference.

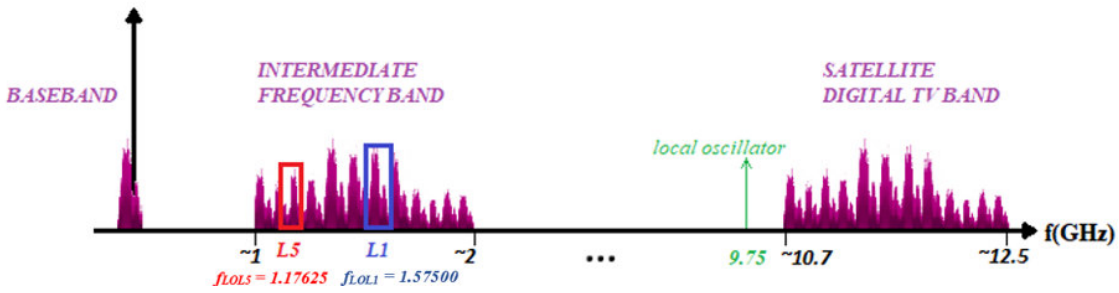


Figure 5.2: Spectrum frequency translation during instrument processing (not in scale).

5.2. Software post-processing

In this section the post-processing techniques that have been applied to the BIBA-SPIR raw data in order to obtain several waveforms will be explained. This signal post-processing has been applied off-line using an open source C++/Fortran90 software library for GNSS-R data analysis and modelling: **`wavpy`** [13]. Some contributions have been made to this library, that will be detailed in section 5.2.3.

The main objective of applying signal processing is to make an instrumental calibration and generate the pulse compression. Figure 5.3 shows a block diagram of the software post-processing of the recorded signals.

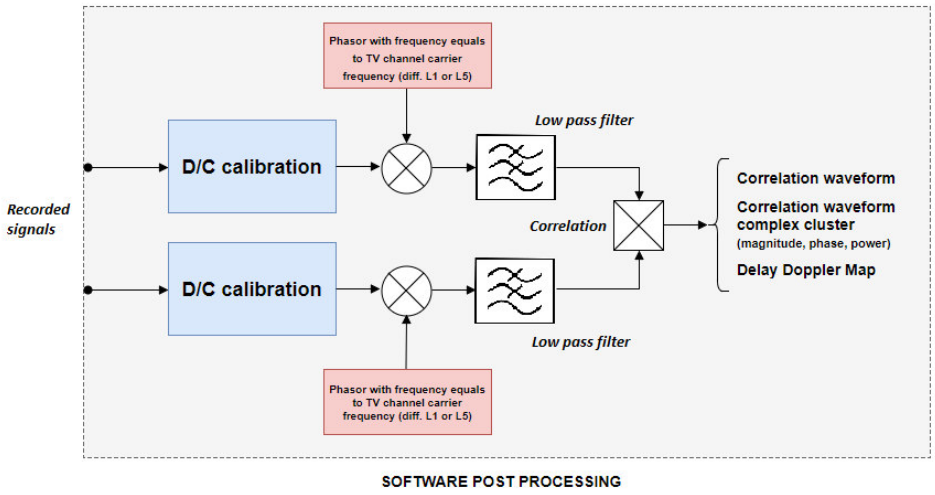


Figure 5.3: Software processing block diagram.

The first step in the processing of the recorded signals is to calibrate the instrumental DC offset originated at the ADC converters. Next, the television channel of the direct signal from which the most power has been received is chosen (depending on the L1 / L5 band and the polarization of the LNB), and the spectrum of the direct and reflected signals is shifted to the base-band by locating the carrier frequency of the channel selected at 0 frequency. Lastly, a digital filter is applied to obtain the selected television channel. This filter is applied in both direct and reflected signals.

The direct and reflected complex signals can be expressed as in equations (5.1) and (5.2).

$$s_{UP}(t) = s(t - T_{UP}) = A(t - T_{UP})\, e^{j2\pi f_c (t - T_{UP})} \tag{5.1}$$

$$s_{DW}(t) = s(t - T_{DW}) = A(t - T_{DW})\, e^{j2\pi f_c (t - T_{DW})} \tag{5.2}$$

Where $s(t)$ is the signal transmitted by the satellite, A is its envelope, f_c is its carrier frequency and T_{UP} and T_{DW} are the time of propagation of the transmitted signal in the direct and reflected path, respectively. Therefore, the complex signals can also be expressed using the in-phase (real) and quadrature (imaginary) components of analytic signals.

$$s_{UP}(t) = i_{UP}(t) + q s_{UP}(t) \tag{5.3}$$

$$s_{DW}(t) = i_{DW}(t) + q s_{DW}(t) \tag{5.4}$$

5.2.1. Correlation

The cross-correlation is defined as a measure of similarity of two signals as a function of its relative displacement. The auto-correlation is the cross-correlation of a signal with itself. The autocorrelation at the origin is always maximum and equals the power of the signal.

In the case of our PBR system, it is desired to determine the bistatic range of the target measuring the time difference (delay) of arrival of the reflected signal respect to the direct signal. The time difference between the reflected signal s_{DW} and the direct path signal s_{UP} can be calculated by cross-correlating these signals. Therefore, the cross-correlation of direct and reflected signals for continuous and discrete time can be computed as (5.5) and (5.6), respectively, in the case of fixed transmitter and receiver antennas and ignoring the effects of moving scatterers (without Doppler).

$$R_{DW,UP}(t_0,\tau) = \frac{1}{T}\int_{t_0}^{t_0+T} s_{DW}(t)s^*_{UP}(t-\tau)\mathrm{d}t \tag{5.5}$$

The asterisk in s^*_{UP} denotes the complex conjugate of the signal. The variable τ is defined as the relative temporal delay applied to the signals. In the signal processing theory, the correlation of two signals is done during an integration time T, in our case we also specify the initial time (t_0) at which the correlation integral is computed. So, we select a segment of the signal of duration T that starts at t_0. When the computation result of the cross-correlation results in a peak, it corresponds to the bistatic time delay.

In practical cases where a receiver device is used, as the BIBA-SPIR receiver in our case, the signals are sampled with a sampling frequency rate f_s. Consequently, the correlation must be computed in discrete-time.

$$R_{DW,UP}(t_0,\tau)\Big|_{t=n/f_s} = R_{DW,UP}[n_0,m] = \frac{1}{N}\sum_{n=n_0}^{n_0+N-1} s_{DW}[n]s^*_{UP}[n-m] \tag{5.6}$$

The computation of cross-correlation of two signals is carried out in the time domain, using the **`wavpy`** [13] software library. Referring to equation (5.6): *N* is the number of samples to be integred, n_0 is the initial sample at which the correlation integral is computed and *m* is the lag number. The output of the cross-correlation computation will result in a complex waveform.

The software range resolution is defined as the lag separation correspondence in distance, in the best case of approximately 3.7474 m, using equation (5.8). This means that if the lag separation is defined as 1 sample, each lag corresponds to approximately 3.7474 m difference distance between UP and DW signals.

$$\begin{aligned} lag_{separation} &= \frac{c}{f_s} \\ &= \frac{299,792,458 m/s}{80 * 10^6 sample/s} = 3.7474 \frac{m}{sample} \end{aligned} \tag{5.7}$$

5.2.2. DC offset Calibration

The analytic complex signals received by the direct path $s_{UP}(t)$ and the reflected $s_{DW}(t)$ at the input of the digital signal processor have real (in-phase) and imaginary (quadrature) components. In an ideal case, the signals $s_{UP}(t)$ and $s_{DW}(t)$ are zero mean, but in a real case there are DC offsets added to the signal as can be seen in equations (5.8) and (5.9).

$$s^m_{UP}(t) = s_{UP}(t) + K_{offset} \tag{5.8}$$

$$s^m_{DW}(t) = s_{UP}(t) + G_{offset} \tag{5.9}$$

$s^m_{UP}(t)$ and $s^m_{DW}(t)$ are the measured signals with an added DC offset. These DC offsets are due to residual components of the noise or instrumental non-linearities (e.g., non-ideal AD converters). In order to make a DC offset cancellation, it is necessary to estimate the average value of these offsets.

$$K_{offset} \approx \tilde{K}_{offset} = \frac{1}{T}\int_0^T s^m_{UP}(t)\mathrm{d}t \tag{5.10}$$

$$G_{offset} \approx \tilde{G}_{offset} = \frac{1}{T}\int_0^T s^m_{DW}(t)\mathrm{d}t \tag{5.11}$$

After computing the approximate values of the DC offset for each signal, they will be subtracted from the measured signals in order to have the ideal and zero mean signals. If the correlation had been calculated without subtracting the DC offset, this would have masked low power signals after cross-correlation.

$$\begin{aligned}
R^m_{DW,UP}(\tau) &= \frac{1}{T}\int_0^T s^m_{DW}(t)s^{m*}_{UP}(t-\tau)\mathrm{d}t \\
&= \frac{1}{T}\int_0^T \left[s_{DW}(t)+K_{offset}\right]\left[s^*_{UP}(t-\tau)+G^*_{offset}\right]\mathrm{d}t \\
&= \frac{1}{T}\left[\int_0^T s_{DW}(t)s^*_{UP}(t-\tau)\mathrm{d}t + \int_0^T s_{DW}(t)G^*_{offset}\mathrm{d}t + \right. \\
&+ \left. \int_0^T s^*_{UP}(t-\tau)K_{offset}\mathrm{d}t + \int_0^T K_{offset}G^*_{offset}\mathrm{d}t\right] \\
&\underset{\text{T is long enough}}{\approx} \frac{1}{T}\left[\int_0^T s_{DW}(t)s^*_{UP}(t-\tau)\mathrm{d}t + \underbrace{\int_0^T K_{offset}G^*_{offset}\mathrm{d}t}_{\text{offset after cross-correlation}}\right]
\end{aligned} \tag{5.12}$$

5.2.3. Filtering

The main function of a filter is to remove unwanted parts of signals, like random noise, or to extract useful parts, such signal components lying within a certain frequency band. In this project, it is desired to implement a filter to extract the maximum possible bandwidth

of a television channel, in order to have a flat spectrum. In this way, we obtain a narrow ambiguity function that is adequate for radar applications.

In the instrument, the analog input signals are sampled and digitalised using analog-to-digital converters, resulting in binary numbers of sampled values of the input signals. In order to perform the signal processing of the sampled signals through software, digital filters will be applied. Some advantages of digital filters over analog filters are listed below:

- Digital filters are programmable, so can be easily modified leaving the hardware unchanged. This flexibility can be exploited on general use computers.
- Digital filters are more stable in terms of time, temperature, humidity. Analog filters are more likely to fail and error, due to the dependency of temperature (particularly if they have active components) and the degradation in time.
- Are not subject to non-linearities (when quantization effects can be disregarded), unlike analog filters.
- When they are software implemented, they can have a high versatility to process signals in a variety of ways, including the ability to be adapted to the changes of input signals.

The main contribution made by me in the library **`wavpy`** [13] has been a new class in C++ called `filter` D.1.1.2., in order to create and define digital filters. Also, filtering functions (`FilteringSignal`) D.1.1.3. have been added in the existing class `complex_signal` to filter the received signals sampled by BIBA-SPIR instrument.

5.2.3.1. Design

Digital filters can be classified according to the necessary data that they need to compute the output. A non-recursive filter, calculates the current output from the current and the previous input values. However, a recursive filter uses previous output values, in addition to input values.

An alternative terminology on filter classification can be done according to the filter impulse response. The impulse response of a digital filter is the output sequence from the filter when an unit impulse (sequence consisting of a single value of 1 at sampling time 0 and followed by zeros in all the remaining instants) is applied at its input. If the impulse response of the filter falls to zero after a finite time, it is known as the Finite Impulse Response (FIR) filter. However, if the impulse response exists indefinitely, it will be known as the Infinite Impulse Response (IIR) filter. Likewise, a recursive filter is known as an IIR filter and a non-recursive filter is known as a FIR filter. We can also have filter with recursive and non-recursive terms simultaneously.

The Fourier Transform of the impulse response of a filter (Figure 5.4) is known as the frequency response of a filter, which tells what the filter output will be (filter and phase gain) at different frequencies.

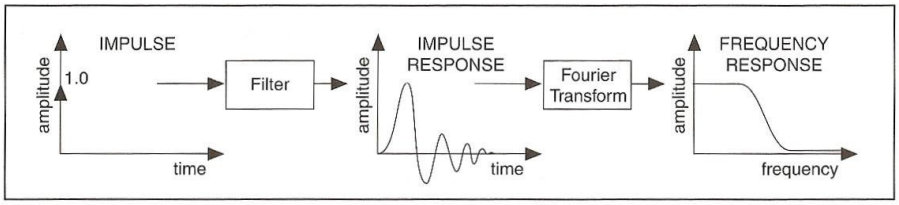


Figura 6.6

Figure 5.4: Impulse and Frequency response of a filter, from [5]

Recursive filters require more calculations to be performed, since there are previous output terms as well as input terms in the filter expression, as will be seen in Section 5.2.3.2.. Nevertheless, to achieve a given frequency response characteristic using a recursive filter generally requires a much lower order filter and therefore fewer terms to be evaluated by the processor than the equivalent non-recursive filter. Overall, with recursive IIR filters, we can generally achieve a desired frequency response characteristic with a filter of lower order than for a non-recursive filter.

Therefore, if IIR recursive filters are implemented, accuracy errors are reduced because of high orders aren't implemented. It must be said that in practical applications, the stable impulse response of the IIR filters decays to near zero in a finite number of samples.

In the experiment carried out, the signals that must be filtered have been previously shifted towards the baseband frequency and subsequently the nearest and with greater amplitude television channel has been centered at the frequency 0 Hz. Then, to extract the selected TV channel a low-pass filter has been applied. The frequency response for the most common IIR filters with the approximation improving as order increases are shown in Figure 5.5.

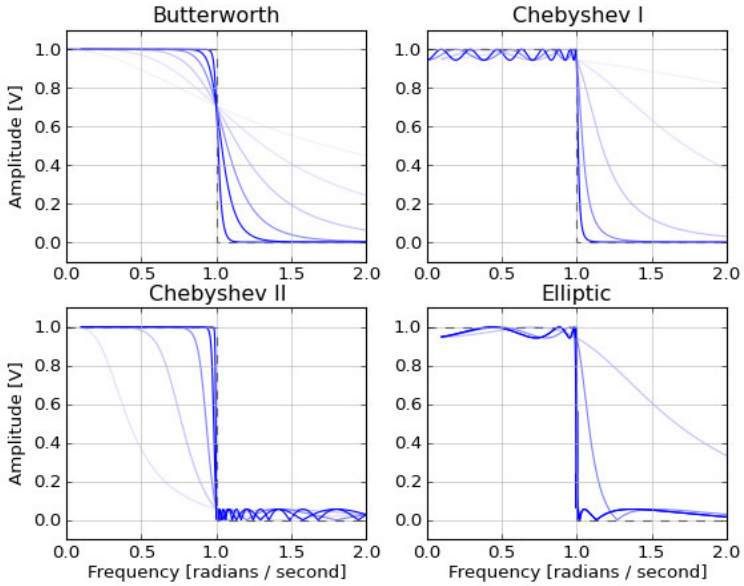


Figure 5.5: Frequency response of common IIR Low Pass Filters, from [5]

For our application, we want a flat amplitude response and near linear phase within the pass-band. In this way, the television channel is wanted to be taken as entirely as possible,

with negligible distortion, and the rest of the frequency spectrum is eliminated. Therefore, the most appropiate filter for the purpose of this work is a Butterworth Low Pass Filter.

The chosen filter has a smooth characteristic response at all frequencies and no ripple, a width of transition region wider than more selective filters and therefore will need a higher order to obtain a sharp transition zone. However, being a programmable filter, permits to specify a cutoff frequency slightly lower than the desired one and keep the filter order low and an acceptable transition zone.

Therefore, observing the spectrum of direct signal in baseband in figures 5.2.3.3. and 5.2.3.3., it was decided to implement a 7^{th} order Butterworth Low Pass filter with f_c = 10 MHz cutoff frequency. This filter optimally extracts the chosen TV channels and gives us an approximately unitary gain, which does not modify the magnitude of the spectrum of the received signals. This filter is implemented having a recursive part and a non-recursive part.

5.2.3.2. Realization

After the design of the digital filter, a structural representation using interconnected basic building blocks is made for the software implementation of the LTI (Linear Time Invariant) digital filter. Therefore, the linear constant-coefficient difference equation can be applied as an algorithm for computing succesive output values as a linear combination of past and present input values and past output values. The signal flow block diagram describes the filter in terms of operation and efficiency, and depending on its structure the filter can be realized on different direct form structures [3].

Taking into account that the filter chosen to be implemented has recursive and non-recursive parts, it can be implemented using the direct form I realization. Its block diagram representation can be interpreted as the cascade of recursive and non-recursive systems, being also visible in its corresponding difference equation (5.13), where both systems are separated.

$$\underbrace{y[n]}_{recursive} = \frac{1}{a_0}\left[-\sum_{k=1}^{N} a_k y[n-k] + \underbrace{w[n]}_{non-recursive}\right] \tag{5.13}$$

$$w[n] = \sum_{k=0}^{N} b_k x[n-k] \tag{5.14}$$

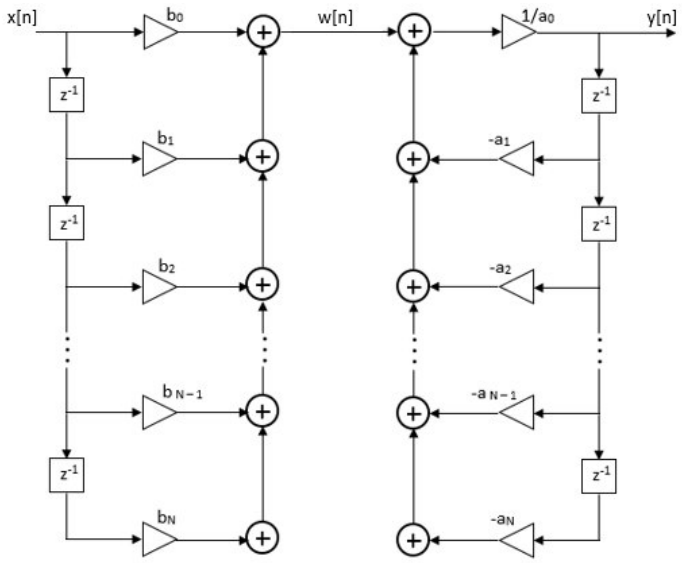


Figure 5.6: Direct form I realization for the LTI system, described by equations 5.13 and 5.14.

It can be seen that the direct form I realization has two delay chains, so it takes twice the memory needed to represent the state of the filter. By reversing the orders of the recursive and non-recursive systems and then collapse both chains of delays into a single chain is obtained the block diagram depicted in figure 5.7. This block diagram corresponds to the Direct form II realization (canonic realization), that can represent the same LTI system as 5.6 with the advantage of needing less delay elements (therefore needing the minimum amount of memory storage).

The difference equation (5.15) that generalizes for a dual recursive and non-recursive LTI system with a single delay chain refers to the Direct form II realization. For more details about the implementation of the code algorithm used for filtering the received signals, the filtering function C++ code is attached in D.1.1.

$$y[n] = \frac{1}{a_0}\left[\sum_{k=0}^{N} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]\right] \tag{5.15}$$

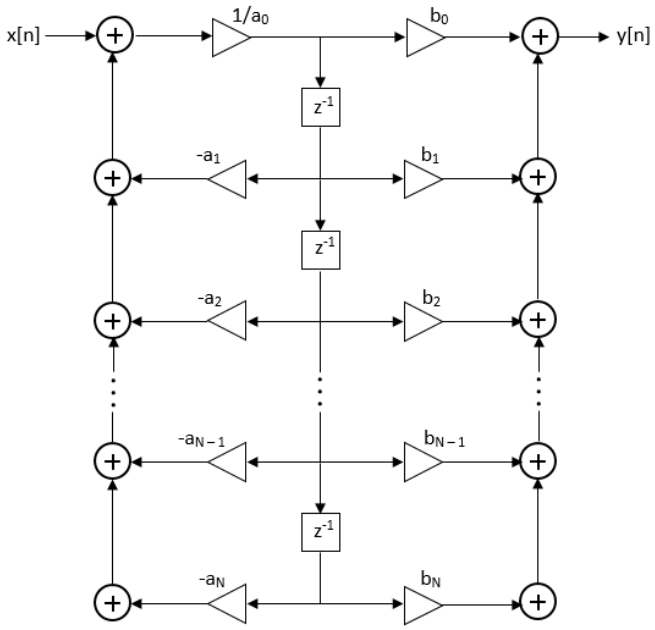


Figure 5.7: Direct form II realization for the LTI system described by equation 5.15.

The implementation of 5.7 is attractive for our application, since very large amounts of data are being processed and the reduction in memory usage also reduces the computation time in the performed calculations in order to filter the signals.

Below an example of software digital filter definition (text file) is shown, where are the parameters used to define a Butterworth filter of order $7^{th}7$ and with cutoff frequency of f_c = 10 MHz.

```
1 7
2 7
3 3.36967377176089e-04 2.35877164023262e-03 7.07631492069787e-03
    1.17938582011631e-02 1.17938582011631e-02 7.07631492069787e-03
    2.35877164023262e-03 3.36967377176089e-04
4 1.00000000000000000 -3.4815187852089329 5.7111518114265420
    -5.5168868794000723 3.3460727800432508 -1.2626357532671739
    0.2728924400139225 -0.0259437893289969
5 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0
```

Where in lines 1 and 2 are defined the order of the Non-recursive and Recursive systems, respectively. In addition, in lines 5 and 6 the initial content of filter real and imaginary taps, respectively, can be observed. The number of complex taps is an indication of the amount of memory required to implement the filter, defined as N = number of delays + 1. In lines 3 and 4 are shown the constant-coefficients for the Non-recursive and Recursive systems, **b** and **a** respectively.

5.2.3.3. Implementation

In this subsection, the results obtained after implementing digital filtering in the received signals (direct and reflected) will be shown.

Following the instrumental processing carried out by the BIBA-SPIR, explained previously in the section 5.1., it is known that depending on the programmed frequency of the local oscillator (different for L1 and L5), the filter of the instrument will extract different TV channels. Then, the television channel of the direct signal from which the most power has been received in each case (L1/L5) will have different central frequency on baseband (power depending on polarization of the LNB).

Therefore, a frequency shifting should be carried out, placing the channel selected for each case at frequency 0 Hz. To perform this action, we must know the carrier frequency of each channel selected for each case and its polarization. The calculations obtained to calculate the center frequency of the selected channels in baseband $f_{TV,BB}$ (before shifting and centering the channel at frequency 0 Hz) are detailed in equation (5.16). To perform the shifting, the signals are multiplied by a phasor of frequency $-f_{TV,BB}$.

$$f_{TV,BB} = f_{TV} - f_{LO_X} - f_{LO_{L_1/L_5}} \tag{5.16}$$

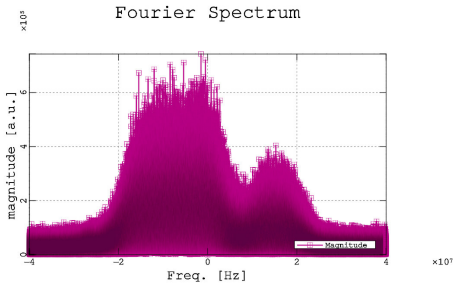
Where f_{LO_X} = 9750 MHz is the instrument local oscillator, $f_{LO_{L_1}}$ = 1575.0 MHz and $f_{LO_{L_5}}$ = 1176.25 MHz are the local oscillator programming frequencies. Taking into account that the LNB had a skew to receive with more power the channels with vertical (V) polarization and knowing an approximate value of the center frequency of the selected channels, the exact value of the carrier frequency for each channel f_{TV} can be estimated looking at the list of TV channels provided by 19.2°E ASTRA satellites. Therefore, the exact values of the center frequency of the selected channels can be computed.

- For L1: f_{TV} = 11318 MHz (Vertical polarization, ASTRA 1KR satellite, DVB-S standard, QPSK).
  $f_{TV,BB}$ = -7.00 MHz
- For L5: f_{TV} = 10936 MHz (Vertical polarization, ASTRA 1M satellite, DVB-S2 standard, 8PSK).
  $f_{TV,BB}$ = 9.75 MHz

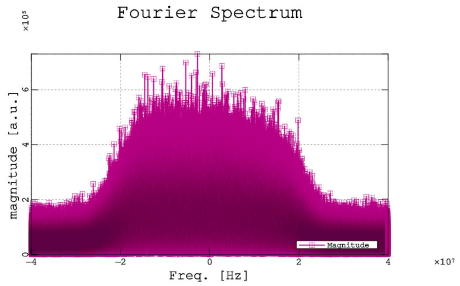
As can be seen, the choice of vertical polarization of the LNB has been chosen so that the filter of the BIBA-SPIR instrument includes most of the received channel with more power within its bandwidth. It was observed that if the LNB was placed with horizontal polarization, the received channels with more power would have central frequencies higher than the filter cutoff frequencies of the instrument and the full channels are not extracted.

Also, it should be noted that vertical polarization gives us the possibility to deal with channels broadcast by different transponders from different satellites and using different angular modulations and standards.

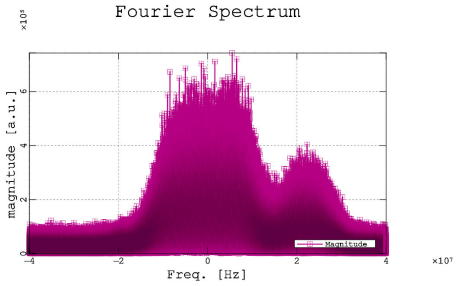
Therefore, in the set of figures 5.2.3.3. and 5.2.3.3. the steps followed for the filtering are shown depending on the frequency L1/L5 chosen. The same procedure is performed for the direct signal and the reflected signal.



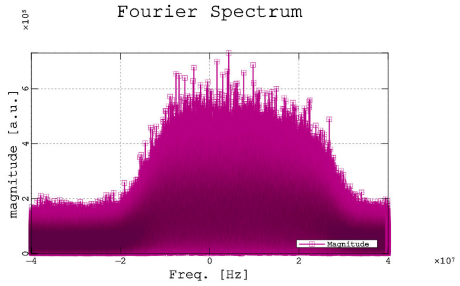
(a) Spectrum of the direct signal before making the shifting to baseband.



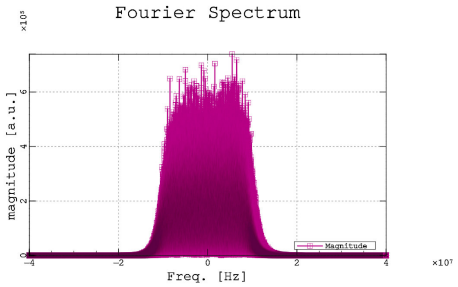
(b) Spectrum of the reflected signal before making the shifting to baseband.



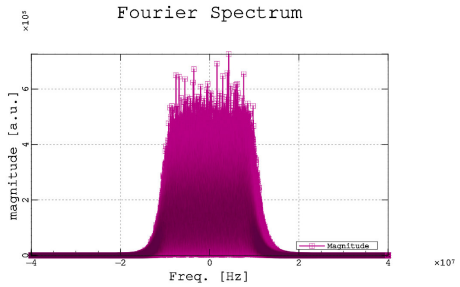
(c) Spectrum of the direct signal after centering the chosen TV channel at frequency 0 Hz.



(d) Spectrum of the reflected signal after being shifted.

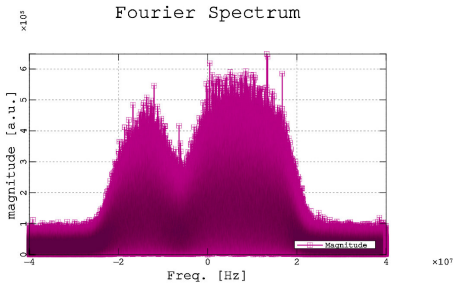


(e) Spectrum of the direct signal after being filtered.

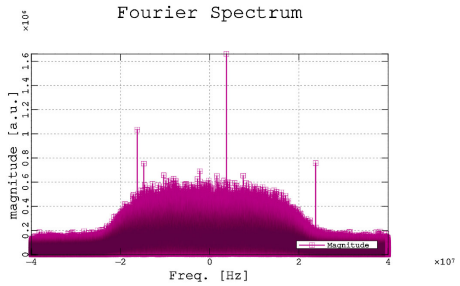


(f) Spectrum of the reflected signal after being filtered.

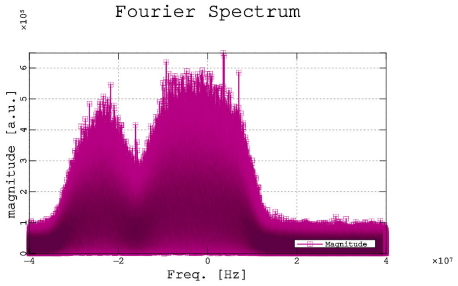
Figure 5.8: Example of digital filtering with a 7^{th} order Butterworth with cutoff frequency f_c = 10 MHz implementation in L1.



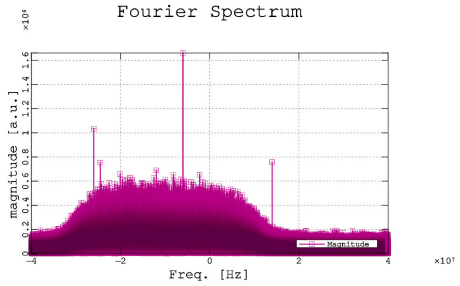
(a) Spectrum of the direct signal before making the shifting to baseband.



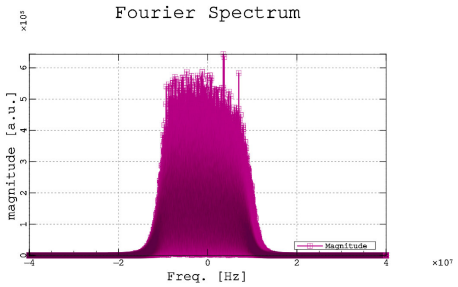
(b) Spectrum of the reflected signal before making the shifting to baseband.



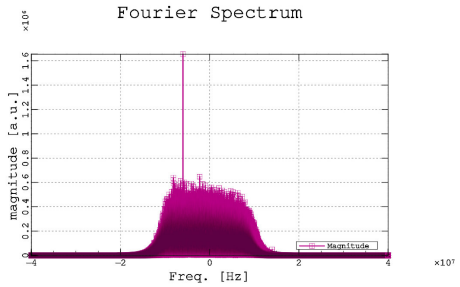
(c) Spectrum of the direct signal after centering the chosen TV channel at frequency 0 Hz.



(d) Spectrum of the reflected signal after being shifted.



(e) Spectrum of the direct signal after being filtered.



(f) Spectrum of the reflected signal after being filtered.

Figure 5.9: Example of digital filtering with a 7^{th} order Butterworth with cutoff frequency f_c = 10 MHz implementation in L5.

Interference tones can be observed at the DW channel in L5, subfigures 5.9(b) 5.9(d) 5.9(f), as the signal strength is lower compared to the UP channel.

5.2.4. Waveform cluster

As mentioned in the section 4.1., the sampled data of the recorded signals is stored into files of 1 GPS second where the I/Q components of the signals are sampled at a rate of 80 Msamples/s. This implies that there is a very high amount of data for each signal, therefore these data is divided into fractions of shorter length and a complex cross-correlation waveform is obtained for each specified window. That is, the number of samples to be integred (N) defines the amount of complex cross-correlation waveforms that will be computed (L)

for each 1-second file.

The complex waveform cluster term appears, which is basically based on the grouping of complex cross-correlation waveforms in a particular period. Thanks to this technique, a graph is obtained where the information provided by multiple complex cross-correlation waveforms can be observed. This is beneficial for the purpose of our project, because if the target to be detected has moved during the second that the receiver has picked up the signals, different bistatic delays will be obtained. When the waveform clusters are computed, the different bistatic time delays peaks can be observed in the same graph (in the case of magnitude and power graphs). In addition, graphs of the phase of the complex waveform cluster can be computed, where the phase variation if the target is in motion can be seen.

In our case, it has been decided to make complex waveform cluster graphs where the information of 999 complex cross-correlation waveforms (L = 999) is included, each calculated with a number of samples to be integred of N = 80000. Some examples can be observed in chapter 6.

5.2.5. Delay-Doppler map

A signal reflected on a moving target will also suffer the effect of Doppler shift wich depends upon the speed and position of the target. Thus, the cross-correlation between the direct and reflected signals will also depend on the Doppler frequency, denoted as f_D, and resulting in a Delay-Doppler map.

The detection of the target is performed by searching the time delayed and frequency Doppler shifted copy of the direct path signal (also known as reference signal) in the reflected signal.

It is known that the cross-correlation between the direct signal and the reflected signal on a moving target will have peaks with maximum signal to noise ratio when the reflected signal is equal as possible to the time delayed version of the direct signal. In order to get these maximum peaks, the direct signal must be corrected with the Doppler frequency of the target. Since the velocity and therefore the Doppler frequency of the target are not known at the receiver, the direct signal must be shifted with all the possible Doppler frequencies. In this way, a frequency scan must be performed. Subsequently, equation (5.17) shows the computation of two-dimensional Delay-Doppler cross-correlation maps.

$$R_{DW-UP}(t_0,\tau,f_D) = \frac{1}{T}\int_{t_0}^{t_0+T} s_{DW}(t)s^*_{UP}(t-\tau)e^{-j2\pi f_D t}dt \tag{5.17}$$

When the signals being examited are discrete, the two-dimensional Delay-Doppler cross-correlation function takes the form of equation (5.18). Where T_s is the sampling time of the instrument.

$$DDM[n_0,m,f_D] = \frac{1}{N}\sum_{n=n_0}^{n_0+N-1} s_{DW}[n]s^*_{UP}[n-m]e^{-j2\pi f_D n T_s} \tag{5.18}$$

5.2.6. Graphic User Interface

Nowadays the Graphical User Interface (GUI) is a popular and widely used method on computer systems whereby information systems interact with users. For this reason, it has been decided to design and implement a GUI for the software processing of the received signals.

The main function of the GUI is to provide the input and output parameters necessary for the correct operation of the *C++* main programs and the used software library that takes care of signal processing. It was decided to create the GUI using a programming language whose syntax favours a readable code and allows the embedding with *C++*, as is *Python*. Specifically, *wxPython* has been used, which is a suite of graphic libraries for Python (programmed in C++) belonging to the *wxWidget* graphic library. *wxFormBuilder* has been used as a a software development tool that simplifies the creation of GUIs by allowing the designer to arrange graphical control elements.

It should be noted that the use of this graphic library allows us to follow the Open Source Initiative philosophy and develop multiplatform graphic applications with portability in *Windows*, *Linux* and *Mac OS X*. The graphical interface baptized as SoOP-GUI (Signals of Opportunity Processing – Graphic User Interface) is shown in Figures 5.10, 5.11 and 5.12.

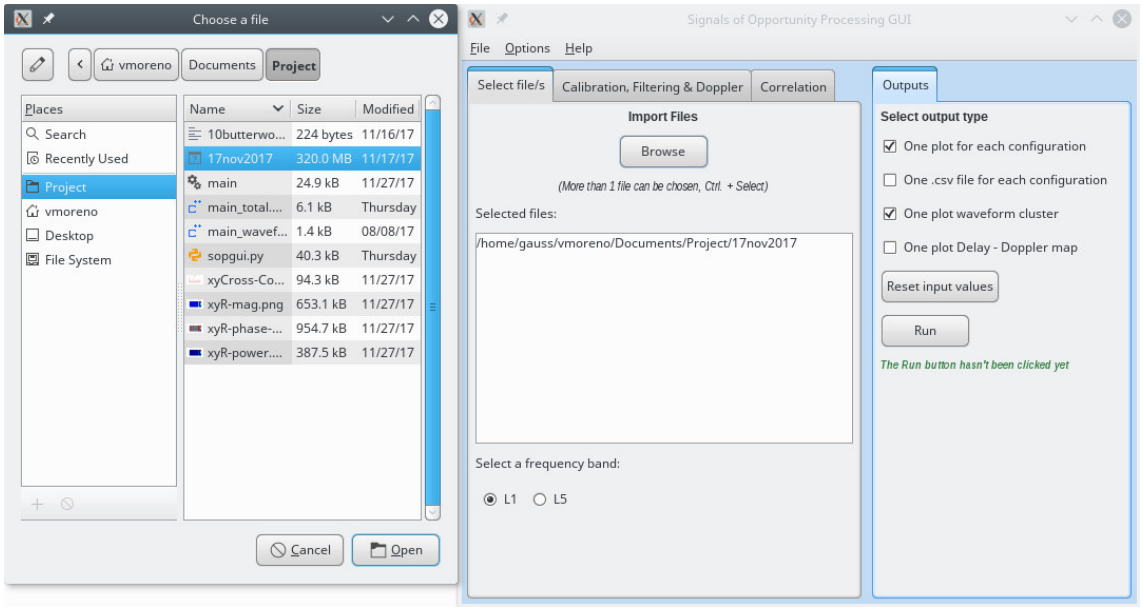


Figure 5.10: Signals of Opportunity Processing – Graphic User Interface [1]

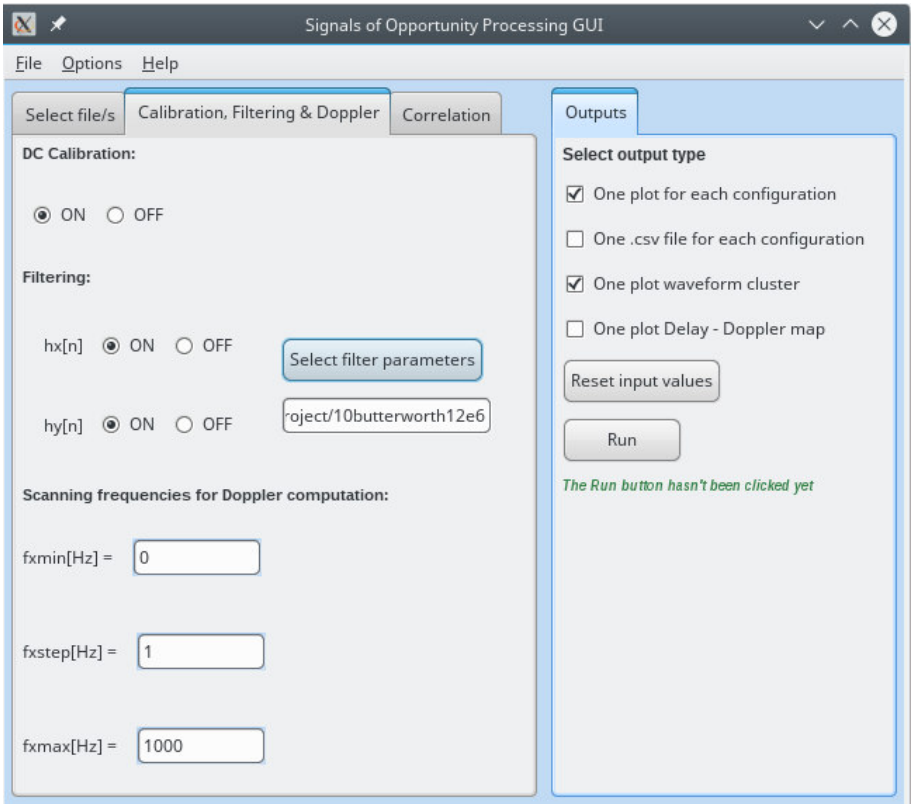


Figure 5.11: Signals of Opportunity Processing – Graphic User Interface [2]

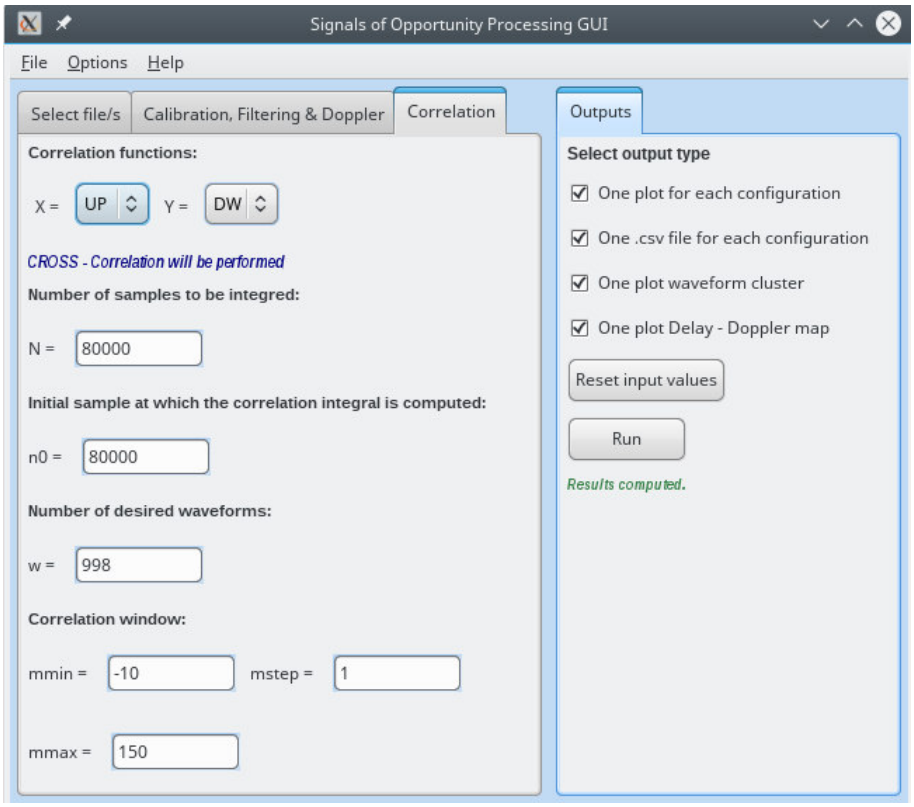


Figure 5.12: Signals of Opportunity Processing – Graphic User Interface [3]

The left part of the graphical interface is dedicated to the input parameters, so there is a Notebook with different tabs. In this section the user can enter data files (signals or filters), correlation parameters, Doppler parameters and even choose whether to perform Calibration and the frequency band. As for the right side of the window, can be found the output options that are wanted as a result of signal processing.

An error control of the files and parameters to be entered by the user has been implemented, displaying error messages if any exceptions are met. Initially the GUI provides some initial parameters by definition, although these can be modified or reseted to the initial values.

CHAPTER 6. EXPERIMENTAL RESULTS

This chapter presents the results from different field experiments carried out to evaluate the performance and capability of the digital satellite TV based passive radar. The results have been divided into the detection of static targets (no movement with respect to the ground station) and the detection of targets with movement, where Delay-Doppler maps can be computed.

All measurements have been made with the UP antenna pointing towards the constellation of 19^{o}E ASTRA satellites, with pointing angles of 155.11^o azimuth and 38.97^o elevation and with Vertical polarization. In order to have a precise pointing for the UP signal, ensuring good SNR, the RF power was monitored using an spectrum analyzer. In each case, the DW antenna has had different pointing angles depending on the location of the target that was wanted to be detected by the radar system.

Due to the location of the system used and having the impediment of not being able to move the entire radar system to any other site, the possible experiments have been carried out, also fulfilling the proposed objectives of range and Doppler detection of the targets. It must be said that many tests have been carried out, but the two that will be shown in this chapter prove the range and Doppler detection of the targets and therefore validate the passive bistatic radar system setup. It should be noted that in the computed graphs, the magnitude and power of the calculated cross-correlations have units of [W] and [W^2], respectively.

6.1. Static target

The first experiment that was carried out was to try to detect a building. A large building has a large BRCS, therefore it is easy to perform the pointing using the directive DW antenna and a fairly high reflected signal power will be received.

In order to have less DSI (Direct Signal Interference), it was decided to point to a building in such a way that the DW antenna did not pick up any component of the direct path signal. Therefore, the building of the Department of Physics of the UAB was the chosen target, as can be observed in Figure 6.1.

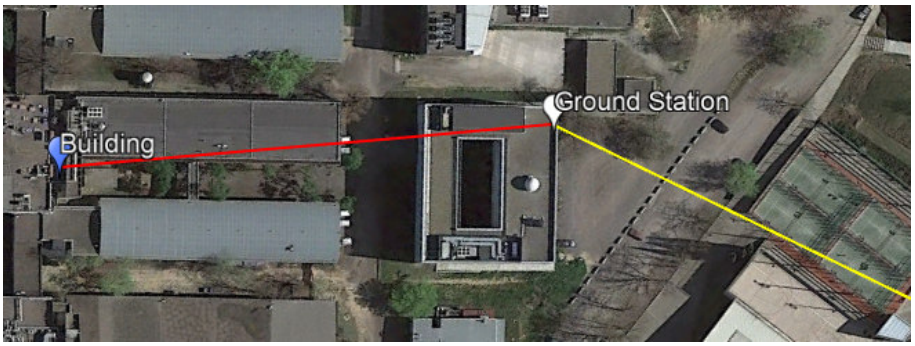


Figure 6.1: Plant view of the ICE building surroundings. Red line: DW antenna pointing azimuth direction, Yellow line: 19.2^{o}E ASTRA pointing azimuth direction.

As explained in previous chapters, lag separation corresponds to approximately 3.75 m. Therefore, having the geographic coordinates of the part of the building that is wanted to be point to, the next step is to choose one altitude at which the reflection of the building will be detected. To avoid interference from other buildings, an altitude of 153 m above the sea level of the reflection point was chosen. In the table 6.1, the pointing angles and the lags at which the building should be detected are shown, among others. Lags correspond to the temporal distance between the UP and DW signals, don't be confused with the distance between the ground station and the target.

Table 6.1: Information about the pointing to the building (computed with a Matlab script).

Name	Lat.[º]	Long.[º]	Alt.[m]	GS-Target distance [m]	Diff. distance [m]	Lag	DW A[º]	DW ε[º]
Ground station	41.500436	2.110422	138	0	0	0	0	0
Building	41.501061	2.109242	153	121.33	193.72	51.66	305.17	6.63

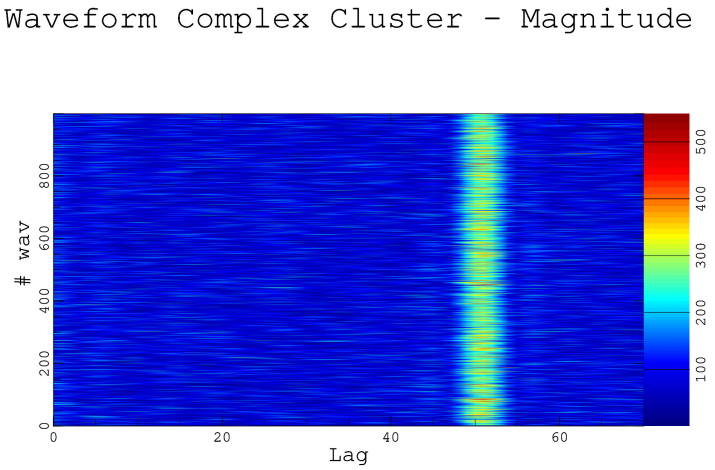
The DW antenna was pointed to the building having an azimuth (A) and elevation (ε) angles of 305.17^{o} and 6.63^{o}, respectively. Assuming the building as a vertical flat plate, the LNB was set into vertical polarization. According to the Matlab code D.3., the difference distance between the direct path signal and reflected path signal corresponds to 51.66 lags.

The next step was to collect the direct and reflected signals with the correct pointing and to perform the signal processing, in order to compute the complex waveform clusters and the Delay-Doppler map. An approximate pointing of 7^{o} of elevation and 305^{o} azimuth was performed.

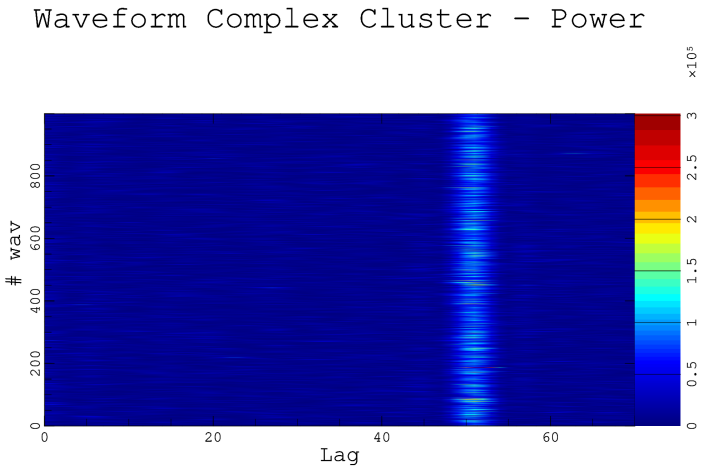
Complex cross-correlation waveform clusters of 1 s, composed of complex cross-correlation waveforms of 1 ms each, have been calculated. As can be seen in the figure 6.2(a) and 6.2(b), approximately at lag 52 a constant peak is observed throughout the cluster that corresponds to the signal reflected off the building. As can be observed in the figure 6.2(c) corresponding to the phase of the complex cross-correlation waveform cluster, the phase is constant during the peak that corresponds to the bistatic delay time lag, because the building does not move with respect to the GS (Ground Station).

When computing the Delay-Doppler for this scenario and performed for 100 ms integration time, as can be seen in the figure 6.2(d), approximately in lag 52 that corresponds to the detection of the building there is a Doppler frequency of 0 Hz. Regarding the frequency resolution of the DDM, increasing the value of N implies that the integration time is longer, so the resolution in frequency improved. In this case, being N = 8e6 (100 ms of time integration) corresponds to a frequency resolution of 10 Hz.

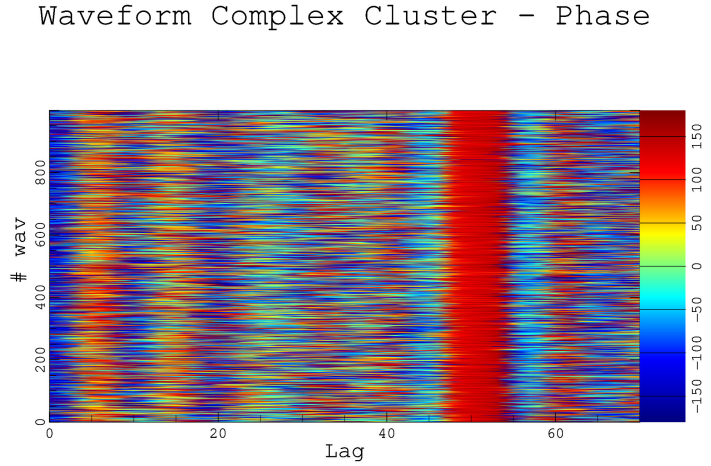
On the whole, the detection of static targets as buildings in our bistatic passive radar system is validated.



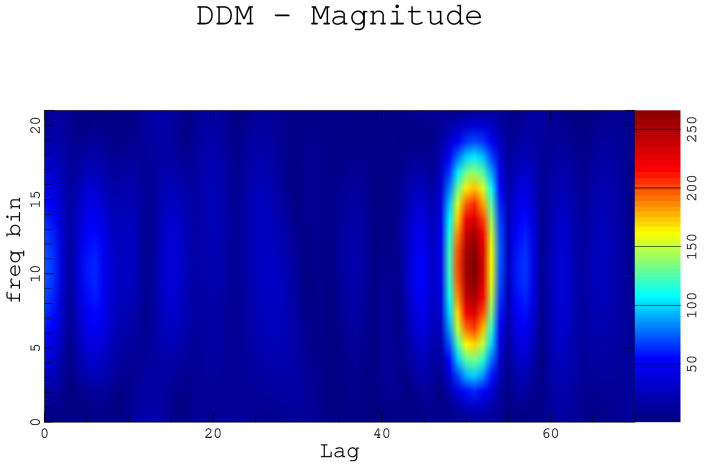
(a) Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(b) Power of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(c) Phase of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(d) Example of DDM (N=8000000 and n_0=32000020)

Figure 6.2: Results of building detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

6.2. Dynamic target

After being successful with the detection of stationary targets, it was decided to try the detection of dynamic targets (with movement in relation to the ground station) where DDM techniques can be applied and the frequencies and Doppler speeds can be computed. It is known that the Doppler signature of a dynamic target is greater for shorter wavelengths, for a given speed. Using digital satellite television signals will provide better results in the calculation of the Doppler effect, compared to using signals with longer wavelength, such as GNSS signals.

An experiment was carried out in which an attempt was made to detect the range and Doppler for a car moving in the parking located behind the institute building, but this did not turn out successful as can be seen in the appendix A.1. The reason therefore was that the possible geometry was with a large bistatic angle and thus with small Doppler signature. The effects of the clutter that was in the parking area where the car was moving are also causes of not getting validated results. Consequently, it was decided to perform the detection of a helium balloon suspended in the air. To have strong reflections at the balloon, it was covered by aluminum foil and in order to have its position semi-controlled it was fastened by fishing lines (see figure 6.2.). Thanks to the bistatic configuration of the balloon experiment and because of the balloon motion with the wind, good results of the Doppler effect were obtained.

Figure 6.3: Ground Station setup with DW antenna pointing to the balloon.

To ensure a good SNR in the balloon detection, a large balloon size of 70 cm diameter was used. Therefore, the BRCS of the balloon is estimated, which is used to characterize by the reflection properties, how detectable an object results for a bistatic radar.

To avoid potential obstruction of the reflected signals by buildings or others, the balloon was elevated to 7 meters of altitude with respect to the ground station. Knowing approxi-

mately the geographical location of the target (see figure 6.4), a Matlab script was used to compute the approximate pointing angles of the directive DW antenna. Using the directive antennato ensure to receive powerful reflected signals and, thus, better SNR but against precise pointing must be achieved. The LNB was set to vertical polarization in order to take the channel with central frequency 11318 MHz.

Table 6.2: Information about the pointing to the balloon (computed with a Matlab script).

Name	Lat.[^{o}]	Long.[^{o}]	Alt.[m]	GS-Target distance [m]	Diff. distance [m]	Lag	DW A[^{o}]	DW ε[^{o}]
Ground station	41.500436	2.110422	138	0	0	0	0	0
Building	41.500476	2.110167	145	22.56	28.89	7.70	281.79	15.42

Taking into account the results obtained in the table 6.2, a directive pointing of the DW antenna with 282^o azimuth and 16^o elevation angles is performed. Considering a static position of the ballon, when computing the cross-correlations waveforms between the direct path signal and the balloon reflected signal, a peak should be observed approximately at lag 7.70, corresponding to the bistatic delay distance.

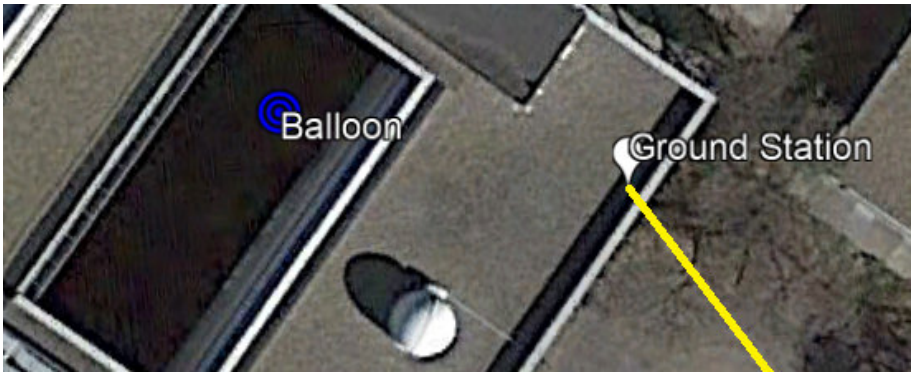
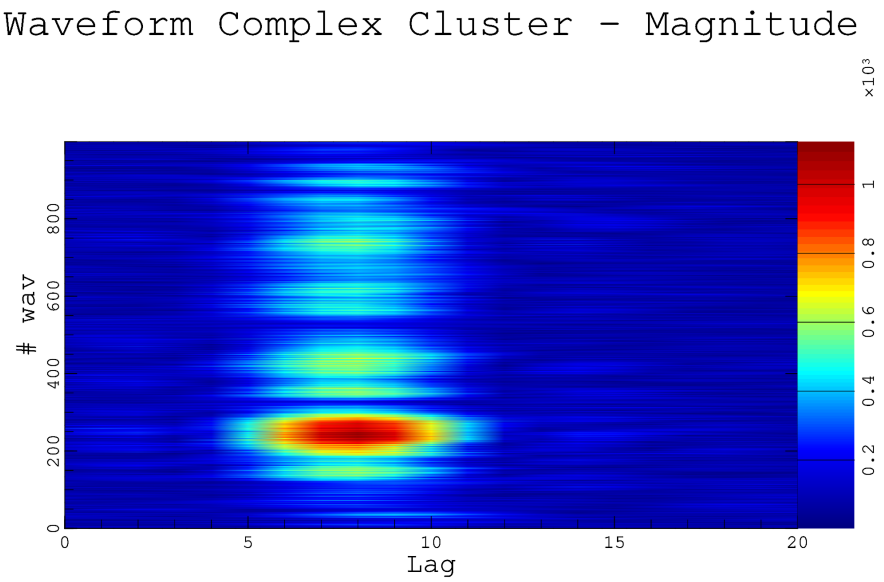


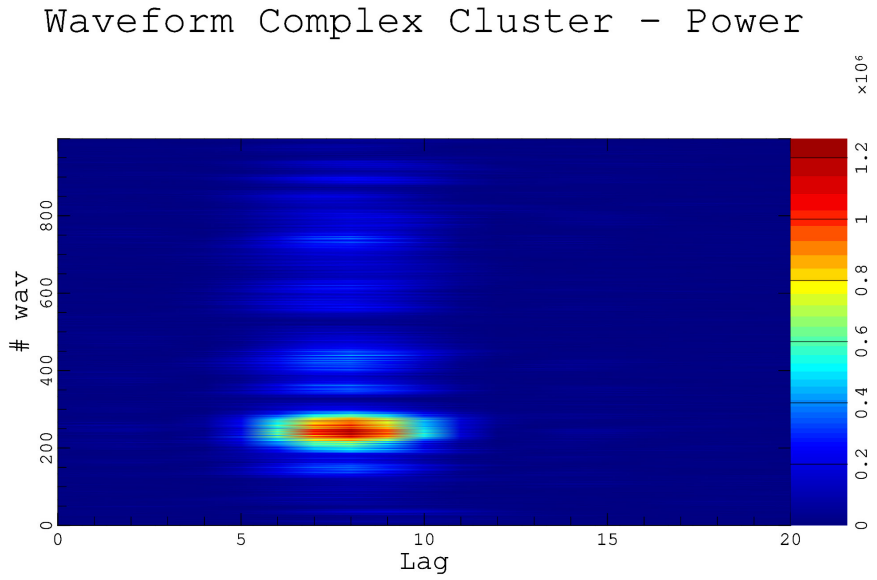
Figure 6.4: Plant view of the ICE building roof. Yellow line: 19.2^{o}E ASTRA pointing azimuth direction.

Complex cross-correlation waveform clusters of 1 s, composed of complex cross-correlation waveforms of 1 ms each, have been calculated and can be observed in figures 6.5(a) and 6.5(b). As can be seen in the graphs mentioned above, it can be observed how the balloon leaves the beamwidth of the DW antenna and is detected only when it is within the detection area. It can be seen that the detection of the balloon occurs in the lag estimated previously by the Matlab script, therefore it is ensured that the detection is from the balloon and not from a different target.

It can also be observed how the magnitude and power of the cross-correlation waveforms varies during the 1 s of the cluster. This also leads us to deduce that the target is in motion.



(a) Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(b) Power of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.

Figure 6.5: Complex cross-correlation waveform clusters of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

The next step to define how the movement of the balloon is to compute the phase graph of the complex cross-correlation waveform cluster. As already mentioned in section 5.2., equations (5.1) and (5.2) expresses how the DW and UP signals depend on their respective propagation time since they are transmitted until they are received at the ground sta-

tion. In the case of detecting a moving target, the propagation time of the DW signal now depends on time. Then, the complex cross-correlation can be expressed as in equation (6.1). Therefore, if the target is moving the phase of the cross-correlation varies depending on the integration time T and if the target is static the phase remains constant. Long integration time gives fine Doppler resolution, and problems due to Doppler walk may arise.

$$R_{DW,UP}(t_0,\tau) = \frac{1}{T}\int_{t_0}^{t_0+T} A(t-T_{DW}(t))A(t-T_{UP}-\tau)^* e^{-j2\pi f[T_{DW}(t)-T_{UP}]}\mathrm{d}t \tag{6.1}$$

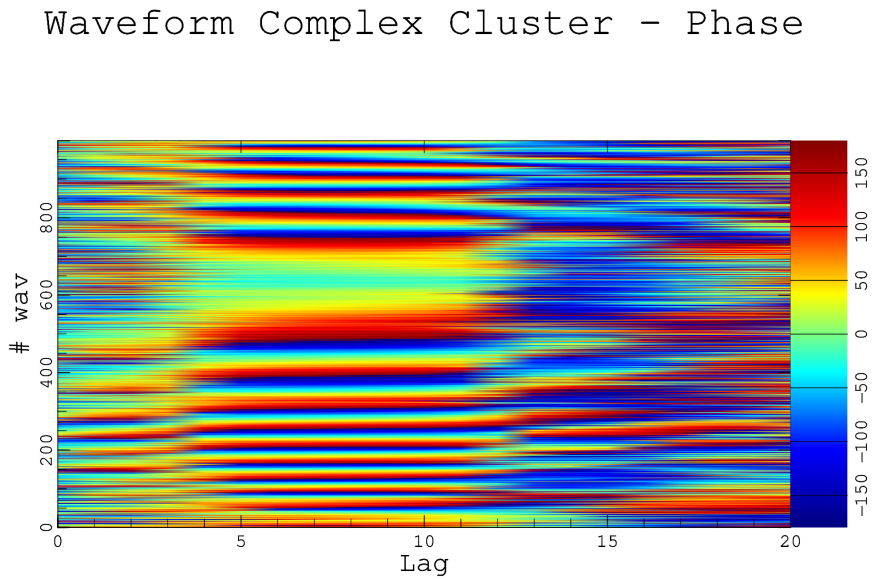


Figure 6.6: Phase of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

As can be observed in figure 6.6, the phase in approximately lag 8 (which corresponds to the detection of the balloon) evolves with negative slope (from -180° to 180°) from waveform 0 to 550 approximately. From waveform 550 to approximately 700 the phase remains more or less constant or with less variations. From approximately the waveform 700 until the end the phase evolves with positive slope (from 180° to -180°), in inverse sense to the initial one. This makes us deduce that the balloon is performing 3 stages of movement in 1 second: first it moves away from the GS, stays more or less static and then approaches the GS.

In order to observe the Doppler effect of the balloon, some Delay-Doppler map graphs have been computed, each of them having 1 ms of integration time. The movement variations of the balloon will be observed during approximately 1 second. Also, it is important to remark that in this case, being N = 8e6 (100 ms of time integration) corresponds to an approximate frequency resolution of 10 Hz ($\frac{80e6}{8e6}$).

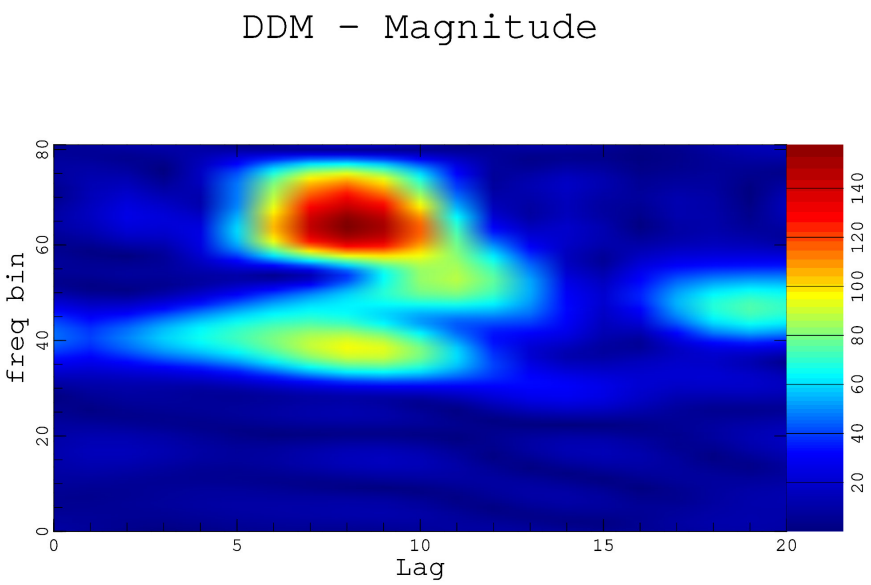


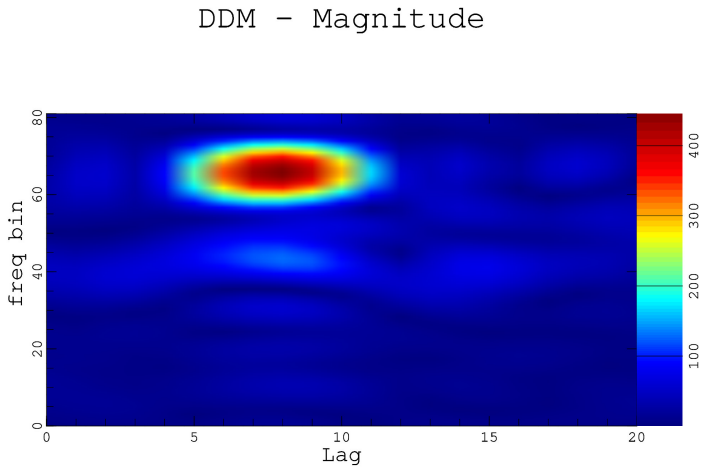
Figure 6.7: First DDM (N=8000000 and n_0=20) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -40 Hz (freq bin 0) to 40 Hz (freq bin 81).

In the figure 6.7, which corresponds to the first instant of 1 ms of the cluster, it can be observed that the balloon is detected in a slight way in its corresponding lag. In the matrix of figures 6.8, a better detection of the balloon can be observed where it has a positive Doppler frequency and therefore it is moving away from the ground station. Subsequently, in the figures 6.9(a) and 6.9(b) the balloon has a Doppler frequency of 0 Hz, therefore it remains approximately static. And in the following graphs 6.9(c) and 6.9(d), it is shown how the balloon now approaches the ground station and therefore its Doppler frequency is negative. In a global way, it can be observed that the lag difference varies slightly between each DDM, that means that the movement of the balloon is small.

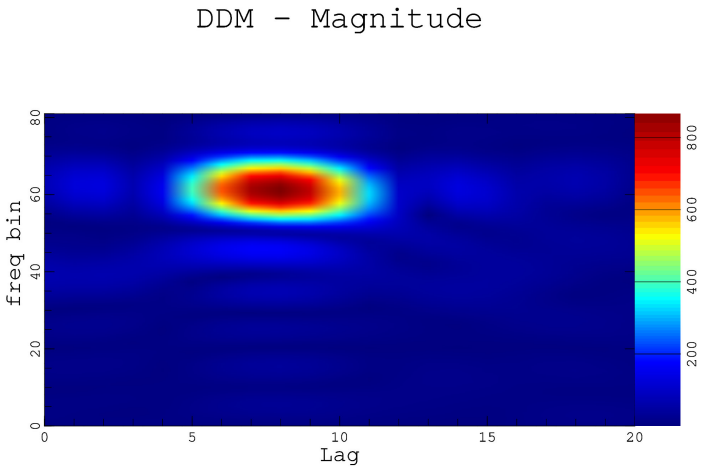
The maximum bistatic Doppler frequency observed in these graphs for the balloon is approximately 24 Hz, that corresponds to a bistatic velocity of 0.318 m/s, using equation (6.2) isolating the velocity from equation (3.11) and taking into account λ = 0.0265 m (for 11318 MHz).

$$v_B = \frac{\lambda f_B}{2} \tag{6.2}$$

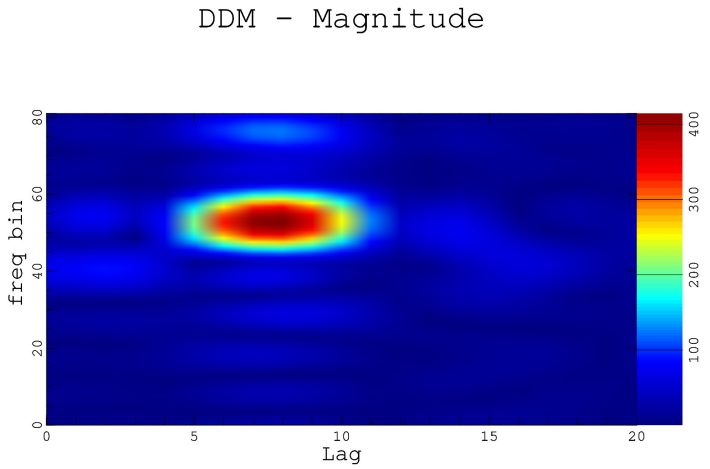
In the appendix A.2., other computed graphs in relation to the movement of the balloon are computed. In these graphs it is shown the motion stages of 1 second of the ballon, before A.2.2. and after A.2.3. the stage analysed in this section. It can be observed how the Doppler frequency related to the movement of the balloon is greater (in both cases) than in the stage shown and the sign of the Doppler frequency is also consistent for both seconds.



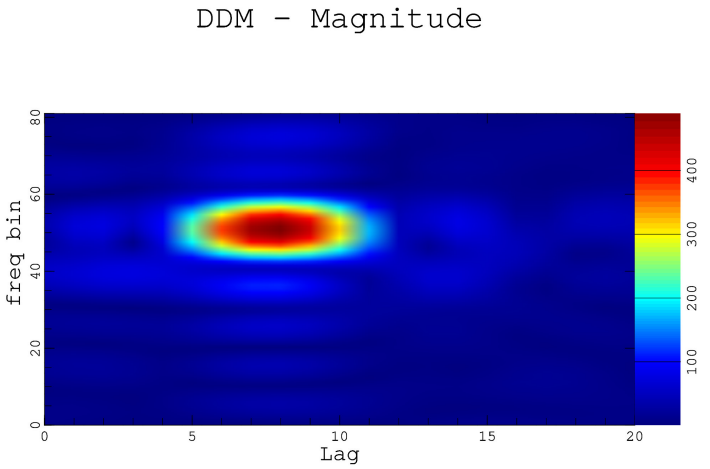
(a) DDM with N=8000000 and n_0=8000020



(b) DDM with N=8000000 and n_0=16000020

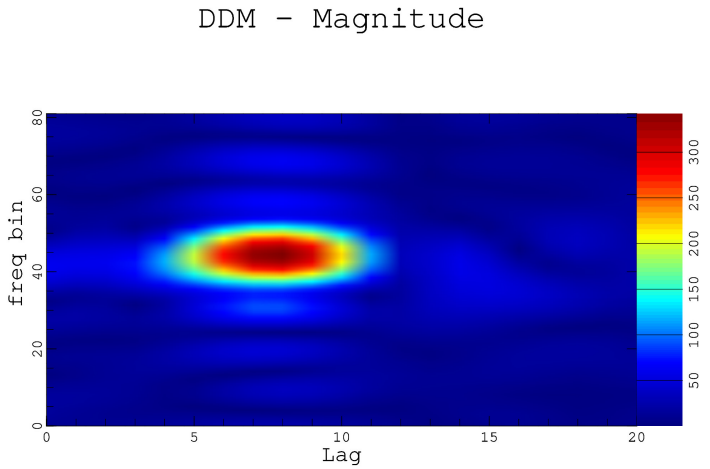


(c) DDM with N=8000000 and n_0=24000020

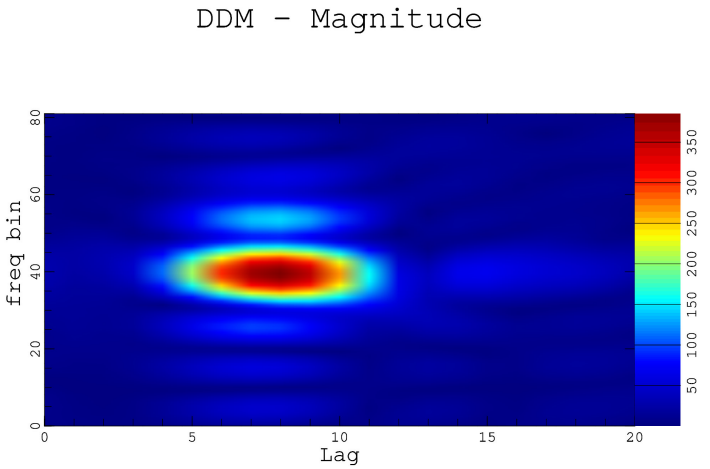


(d) DDM with N=8000000 and n_0=32000020

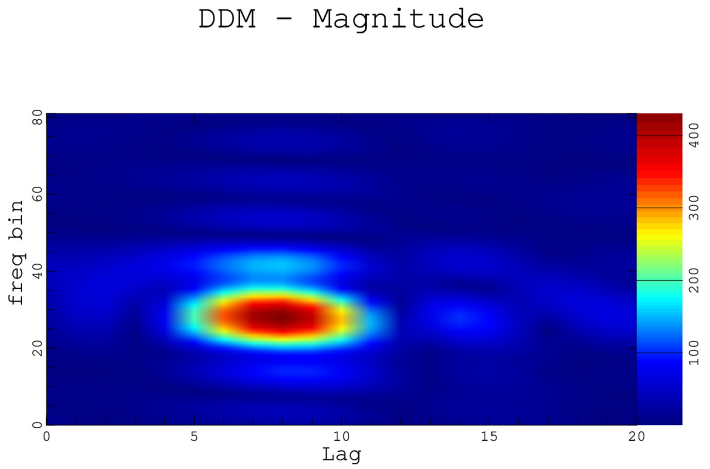
Figure 6.8: Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (first part) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -40 Hz (freq bin 0) to 40 Hz (freq bin 81).



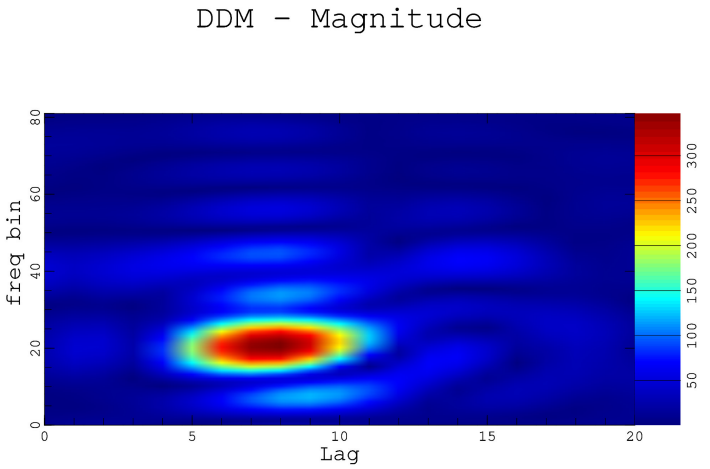
(a) DDM with N=8000000 and n_0=40000020



(b) DDM with N=8000000 and n_0=48000020



(c) DDM with N=8000000 and n_0=56000020



(d) DDM with N=8000000 and n_0=64000020

Figure 6.9: Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (second part) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -40 Hz (freq bin 0) to 40 Hz (freq bin 81).

CONCLUSIONS & FUTURE WORK

Conclusions

This project has described a passive bistatic radar concept using digital satellite television signals as signals of opportuniy for detecting range and Doppler properties of different targets.

Firstly, research tasks were carried out where the work of scientific and technical researchers in the field of radar using signals of opportunity was analysed. Subsequently, a study of the potential of satellite digital television signals has been carried out for its use as signals of opportunity in bistatic radars. The bistatic configuration of the PBR and the properties of the opportunity signal used have been key factors that define the potential of our radar system.

One the one hand, a ground station has been assembled formed by two receiving antennas (one for each link) and using the existing receiver instrument called BIBA-SPIR, in charge of recording and sampling the signals received by both antennas. During the accomplishment of these tasks, knowledge related to electronics and telecommunications has been learned.

On the other hand, the central part of the project is based on off-line software signal processing and the implementation of new code to the library `wavpy`. Techniques such as the cross-correlation between signals in order to measure the distance delay between the direct signal and the reflected signal have been studied. Among others, the use of digital filters in the library has been implemented. As an extra touch to the project, a graphical interface that provides visual support has been implemented in the software signal processing.

Finally, different experiments have been carried out to prove the validation of the PBR system to detect targets. These have demonstrated the capability of the proposed passive bistatic radar concept to detect targets and measure their delay and Doppler signatures. As a result of the investigation, an abstract has been prepared for the 2018 IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2018), to be held on the July 23–27, 2018.

It must be said, that initially the main objective of the project was to detect precipitation. Due to natural factors, precipitation events with strong intensity have not occurred near the ground station and therefore, in advance, it was decided to perform different target detections. The detection of rain will be considered as a possible application in future works. The development of this project, has allowed to see that the implementation of passive bistatic radars using digital satellite television signals is feasible for target range-Doppler detection.

In conclusion, as far as I'm concerned, the realization of this project has been a challenge for me, it has made me to acquire new knowledge and it has given me values that enrich me as a professional and as a person.

Future work

In order to improve the detection of targets with our PBR system in future works, it is necessary to mention different aspects to implement new techniques that could be incorporated into the existing system or the to the software processing signal library.

Digital satellite television signals are transmitted alternating frequency channels with vertical and horizontal polarization. In this project, the channels that were received with more power have been filtered, which in most cases used to be with vertical polarization. The fact of having an LNB with a single output and a single polarization, limited us to not being able to receive both polarizations simultaneously. Therefore, the optimal solution to this problem is to install LNBs with several outputs and thus be able to implement a part of software signal processing that chooses the channel that has been received with greater power.

In some bistatic configurations, the direct signal incoming at the DW antenna interfere the reflected signals and also leaks into the secondary lobes of the antenna that points to the scatterers. This may hide weaker echoes that are below the secondary lobes of the ambiguity function. Then, in order to supress the direct signal interference, some mitigation techniques algorithms should be used in the software signal processing. In the used software library, there is a recent new incorporation that is responsible for the DSI mitigation. Also, this direct path interference could be solved by the application of phased array antenna systems.

Another important aspect is that the signal received from the target and clutter often go through multipath unknown channels. Therefore, another future work will be to estimate the clutter signal components [20] in order to cancel them.

There are multiple digital satellite television transmitter that are illuminating the Earth. This redundancy offers the possibilty to combine multiple transponders, therefore increasing the bandwidth, leading to an improved range resolution of the passive bistatic radar.

Moreover, in future campaigns the Doppler signatures of the ocean currents will be measured. There is still a need to demonstrate rain detection using passive bistatic radar, for which various software signal processing techniques are being developed.

Finally, a portable version of the radar used could be made and incorporated into a van to be able to go to areas where precipitation events with high intensity frequently occur. Also, a variety of experiments could be performed without being limited by the current environment in order to check the potentialities of the used passive bistatic radar.

BIBLIOGRAPHY

[1] *IEEE Standard Radar Definitions*, 686-2008, 2008.

[2] Nicholas J. Willis, Hugh D. Griffiths. *Advances in Bistatic Radar*, 1995. 2007.

[3] Oppenheim Alan V, Willsky Alan S. Linear Time-Invariant Systems. *Signals and systems* (pp. 111-117), 1999.

[4] Ching Wei Wesley Chang. *Television Based Bistatic Radar, System Level Investigations of Television Based Bistatic Radar*, 2010.

[5] Tutorial de Adquisición y Procesado de Señales en LabVIEW, Universidad del País Vasco. Retrieved from http://www.ehu.eus/procesadoinsvirtual/Contenido.html

[6] Dagoberto Salazar. "Navegación Aérea, Cartografía y Cosmografía", *EPSC-UPC*, 6ta versión, 2008.

[7] Balázs Lábsky and Tomáš Kratochvíl, "DVB-S/S2 Satellite Television Broadcasting Measurement and Comparison," *2010 20th International Conference Radioelektronika*, 2010.

[8] Zeyue Sun, Tianyun Wang, Tao Jiang, Chang Chen, Weidong Chen, "Analysis of the Properties of DVB-S Signal for Passive Radar Application," *Wireless Communications & Signal Processing (WCSP), 2013 International Conference*, Oct 2013.

[9] M. Martín-Neira, S. D'Addio, C. Buck, N. Floury, and R. Prieto-Cerdeira, "The PARIS Ocean Altimeter In- Orbit Demonstrator," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 49, no. 6, pp. 2209 –2237, june 2011.

[10] M. Martín-Neira, "A passive reflectometry and interferometry system (PARIS): Application to ocean altimetry," *ESA J.*, vol. 17, no. 4, pp.331–355, Dec. 1993.

[11] Serni Ribó, Juan Carlos Arco-Fernández, Estel Cardellach, Fran Fabra,Weiqiang Li, Oleguer Nogués-Correig, Antonio Rius, and Manuel Martín-Neira, "A Software- Defined GNSS Reflectometry Recording Receiver with Wide-Bandwidth, Multi-Band Capability and Digital Beam-Forming," *Remote Sensing*, vol. 9, no. 5, 450, 2017.

[12] Serni Ribó, Changhui Jia, Juan Carlos Arco-Fernández, Weiqiang Li, Estel Cardellach, Fran Fabra, and Antonio Rius, "The Bi-Band Software PARIS Interferometric Receiver," May 2017.

[13] F. Fabra, E. Cardellach, W. Li, and A. Rius, "Wavpy: A GNSS-R open source software library for data analysis and simulation," *in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2017, pp. 4125–4128.

[14] S. Ribó, J. C. Arco, S. Oliveras, E. Cardellach, A. Rius, and C. Buck, "Experimental Results of an X-Band PARIS Receiver Using Digital Satellite TV Opportunity Signals Scattered on the Sea Surface," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 9, pp. 5704–5711, Sept 2014.

[15] M. Radmard, M. Bastani, F. Behnia, M. M. Nayebi, “Advantages of the DVB-T Signal for Passive Radar Applications,” *Radar Symposium (IRS), 2010 11th International Vilnius*, Lithuania, 16-18 June 2010.

[16] Tamás Peto, Levente Dudás, Rudolf Seller, “DVB-T Based Passive Radar,” *Radioelektronika, 2014 24th International Conference, IEE transactions*, Slovakia, 2014.

[17] Stefan Brisken, Matteo Moscadelli, Viktor Seidel and Christoph Schwark, “Passive Radar Imaging Using DVB-S2,” *Radar Conference (RadarConf), 2017 IEEE*.

[18] R. Shah, J. Garrison, M. Grant, S. Katzberg, and G. Tian, “Analysis of the correlation properties of digital satellite signals and their applicability in bistatic remote sensing,” in *Proc. IEEE IGARSS,* 2010, pp. 4114–4117.

[19] R. J. Watson, B. Wiltshire, N. Dumont. “Measurement of atmospheric refractivity using signals of opportunity,” in *Proc. IEEE IGARSS,* 2012.

[20] F Colone, R Cardinali, P Lombardo, “Cancellation of clutter and multipath in passive radar using a sequential approach,” *Proceedings of IEEE International Conference on Radar*, IEEE, Verona, 2006.

[21] F Colone, DW O’Hahan, P Lombardo, CJ Baker, “A multistage processing algorithm for disturbance removal and target detection in passive bistatic radar,” IEEE Trans. Aerospace Electron. Syst.45:, 698–722 (2009).

[22] A Ansari, MR Taban, “Implementation of sequential algorithm in batch processing for clutter and direct signal cancellation in passive bistatic radars,” *Proceedings of Iranian Conference on Electrical and Electronic Engineering (ICEE2013)*, Mashhad, 2013.

[23] Hilde Kjelgaard Brustad, “Direct signal cancellation in passive bistatic DVB-T based radar,” Forsvarets forskningsinstitutt. FFI. Norwegian Defence, 2014.

[24] Christopher Coleman, “ Mitigating the effect of direct signal interference in passive bistatic radar,” *Radar Conference - Surveillance for a Safer World, 2009. RADAR. International*, March 2010.

[25] Zeinab Shamaee, Mohsen Mivehchy, “RFIA: A Novel RF-band Interference Attenuation Method in Passive Radar ,” *Journal of Electrical and Electronic Engineering*, May 2016.

[26] Joseph Landon Garry, Graeme Edward Smith, C.J. Baker, “ Direct signal suppression schemes for passive radar,” *2015 Signal Processing Symposium (SPSympo)*, June 2015.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

INSTITUT DE
CIÈNCIES
DE L'ESPAI
CSIC IEEC

APPENDICES

TFG TITLE: Validation of a Passive Bistatic Radar using Digital Satellite TV as a Source of Opportunity

DEGREE: Bachelor's Degree in Air Navigation Engineering

AUTHOR: Víctor Moreno Rodríguez

ADVISOR: Serni Ribó Vedrilla

SUPERVISOR: Jordi Berenguer i Sau

DATE: February 6, 2018

APPENDIX A. OTHER EXPERIMENTAL RESULTS

A.1. Car detection

With this experiment, we tried to detect the range and speed of a car moving through the parking located behind the building of the Institute of Space Sciences. For this, the UP antenna was pointing towards the constellation of 19^{o}E ASTRA satellites, with pointing angles of 155.11^{o} azimuth and 38.97^{o} elevation and with Vertical polarization. In this case, we wanted to use the non-directive antenna, using only the feedhorn without the antenna dish, to have much broader beam and obtain more data seconds of the DW signals reflected in the car while it is moving.

In order to know what is the second exact GPS in which the car was in the coverage volume of the DW antenna and to be able to extract the data collected by the BIBA-SPIR instrument at this time, a Topcon GPS receiver located in the car with a sampling rate of a position measurement every second was used. In the figure A.1, you can see the car's path every 1 second and the coverage area of the antenna DW. As for the colors: black means parked car, yellow means that the car is performing maneuvers and the rest colors show different trajectories. To obtain some information about the car trajectory using the Topcon positioning system, some configuration and fyle type changes was made to RINEX. This information was sent to a Canadian on-line server that provides Precise Point Positioning information in a pdf file C.

It can be seen that in points A and B, green path, the car has higher speed than in the other trajectories and there are 2 positions within the coverage area of the antenna. Therefore, these points are selected to check the detection of the car from the reflection of digital satellite television signals.

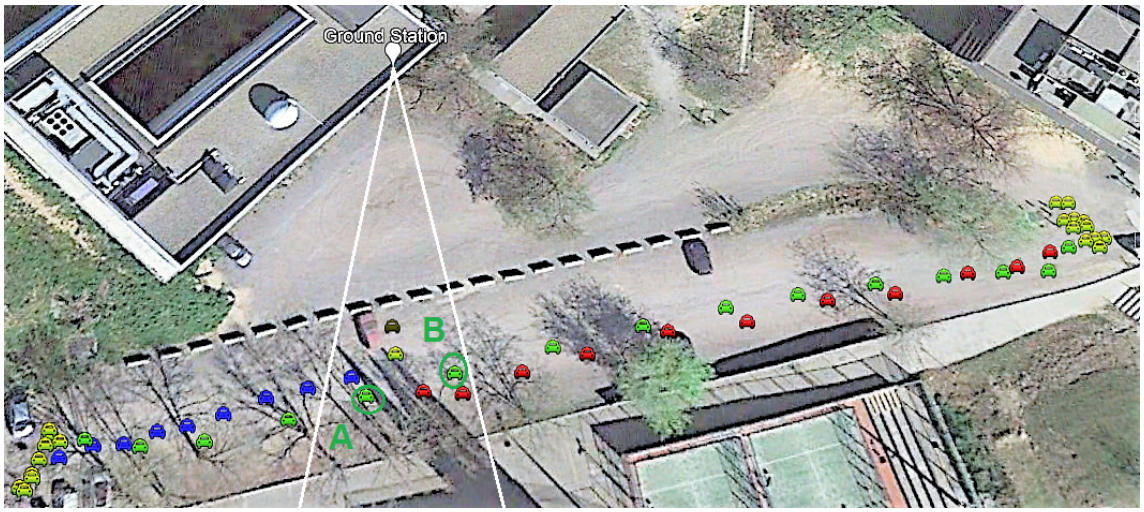


Figure A.1: Car trajectory estimation using *Topcon* GPS receiver (sampling rate of a position measurement per second), using non-directive DW antenna.

There have been different factors that are possible causes of the non-validation of this experiment. The bad bistatic geometry that exists in this experiment makes the Doppler effect not detectable, since the car moves almost perpendicularly to the GS. It should have been chosen another car trajectory, but we were limited by the surroundings of the building and that was the only option to have the car with considerable speed. Another important factor that has influenced the detection of the car is the fact that the car park was full of cars, thus there is a high amount of reflections of digital satellite television signals. The fact of having used the non-directive DW antenna gave more coverage, but the recorded reflected signal of the car in motion is weaker and other reflected signals were also received. The DW directive antenna could have been used, but the pointing towards the car in motion would have been very difficult to perform.

The Matlab script predicts that the range detection of the car in both positions (A and B) must be close to lag 5.

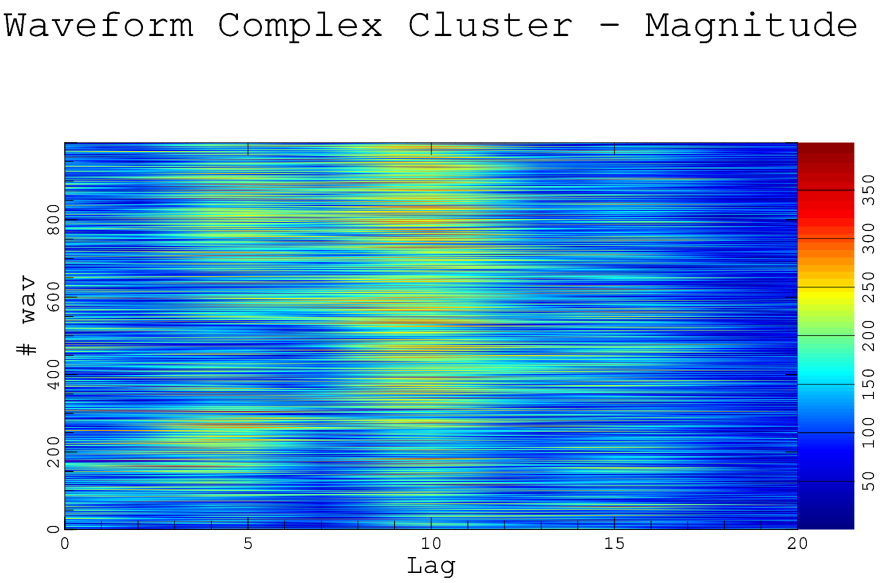


Figure A.2: Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms, each for position A. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

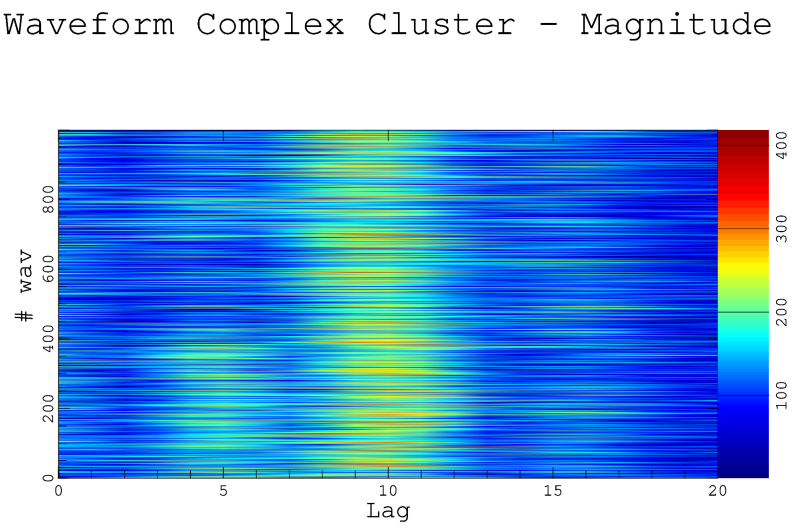


Figure A.3: Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each, for position B. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

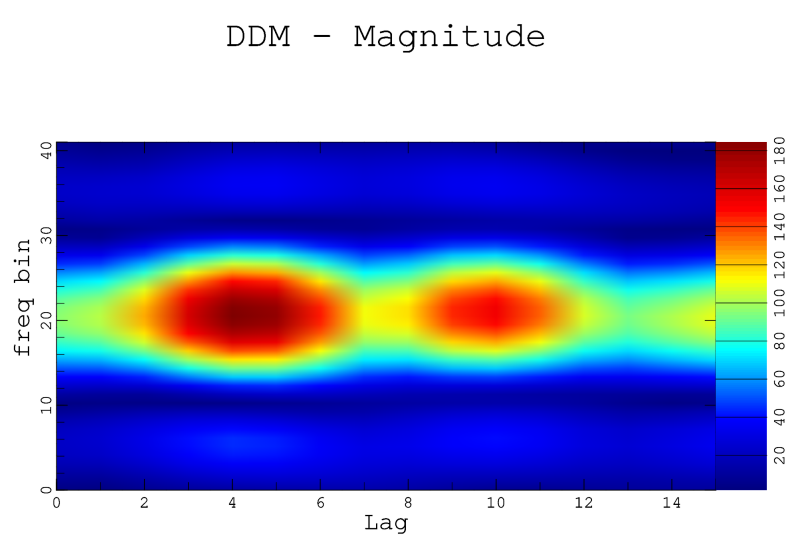
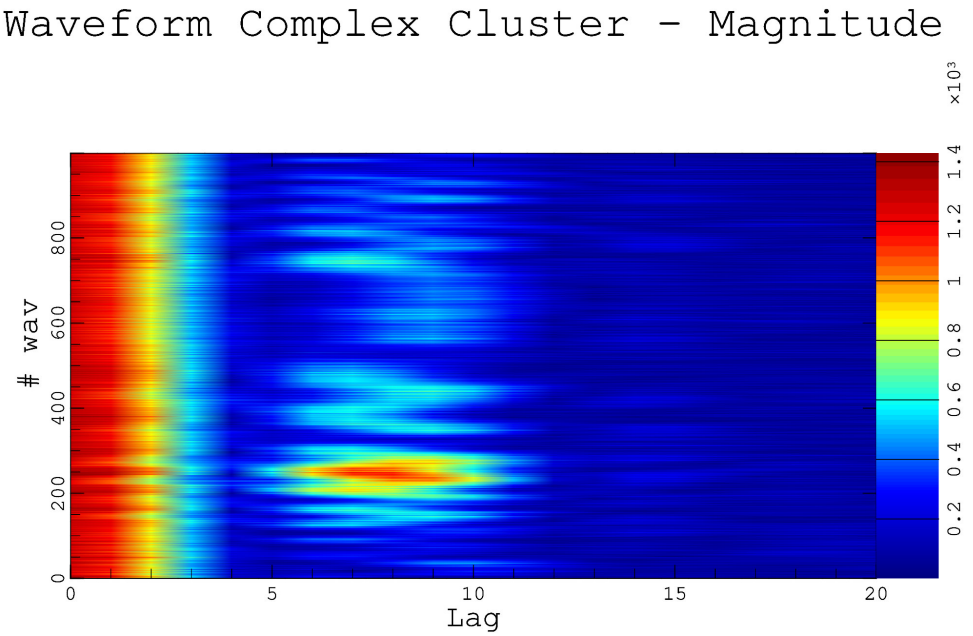


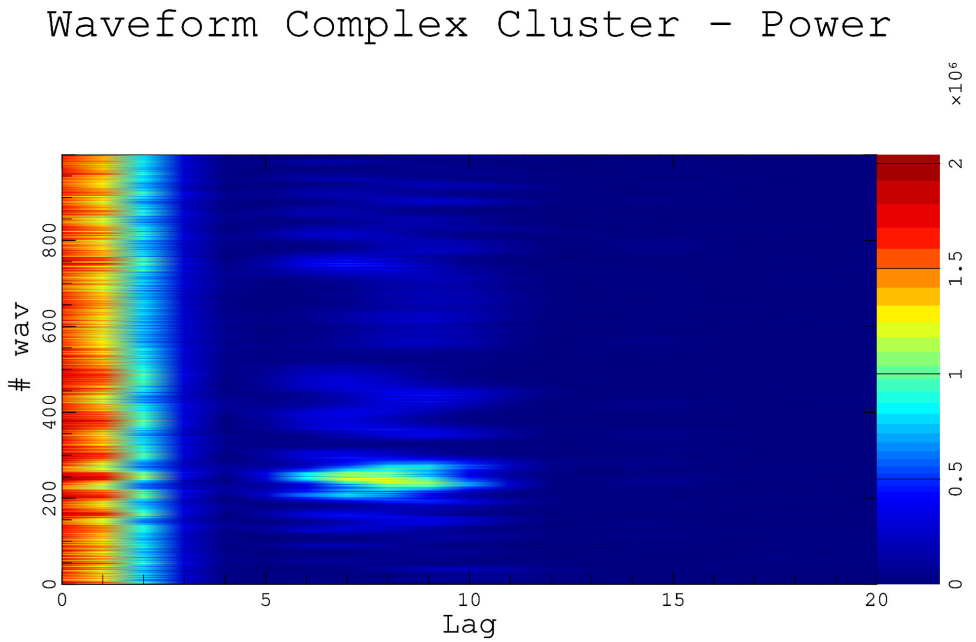
Figure A.4: Example of DDM of 1 ms with N=8000000 and n_0=16000015, for A.3. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -20 Hz (freq bin 0) to 20 Hz (freq bin 41).

A.2. Balloon detection

A.2.1. Ballon detection shown without DSI mitigation



(a) Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(b) Power of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.

Figure A.5: Complex cross-correlation waveform clusters of Balloon detection without DSI mitigation. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

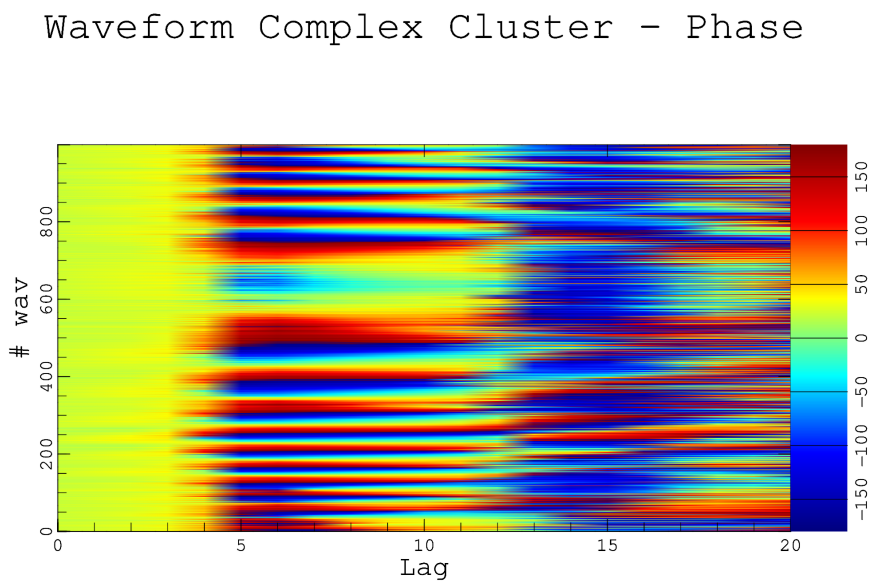


Figure A.6: Phase of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each. Balloon detection without DSI. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

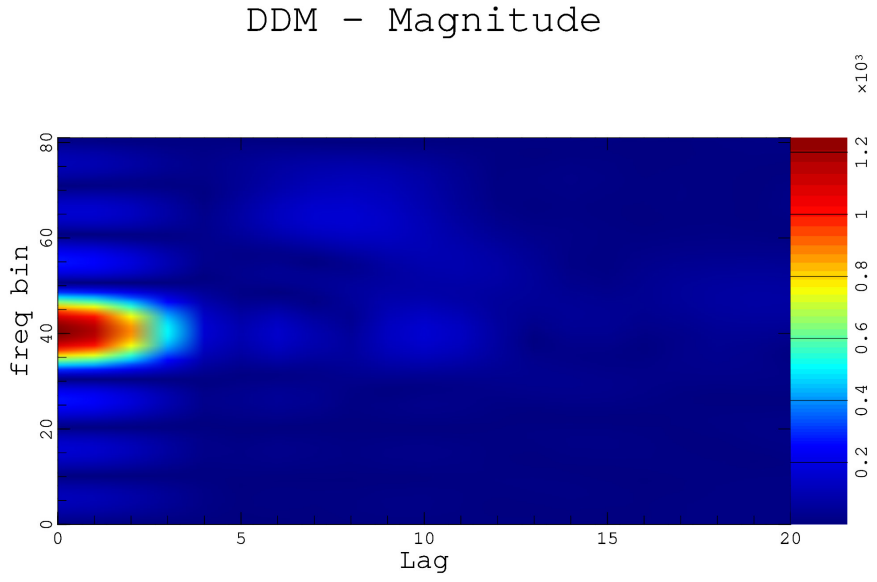
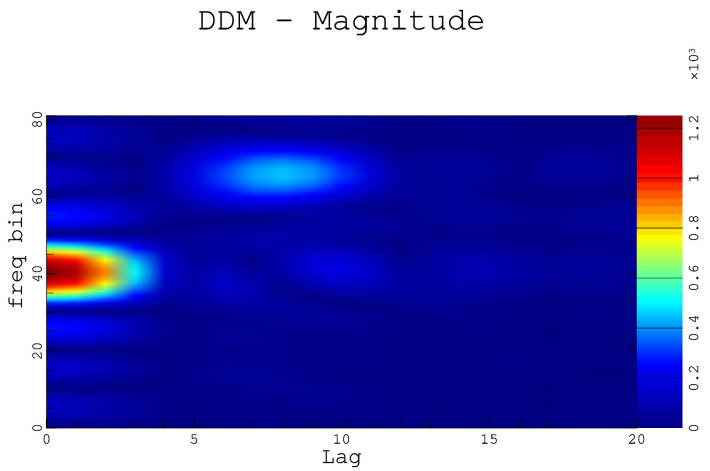
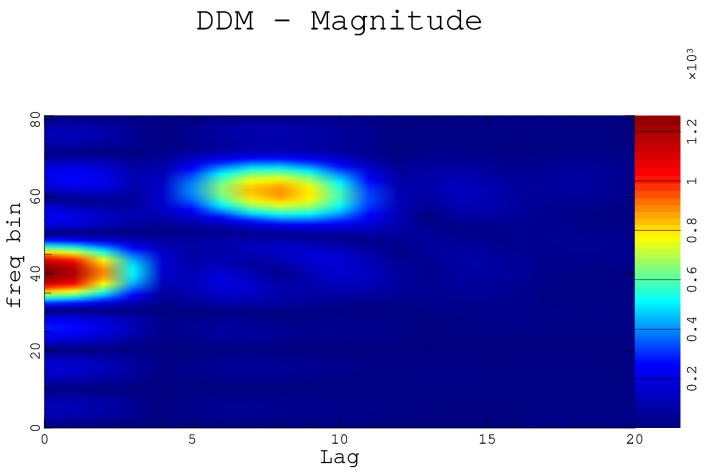


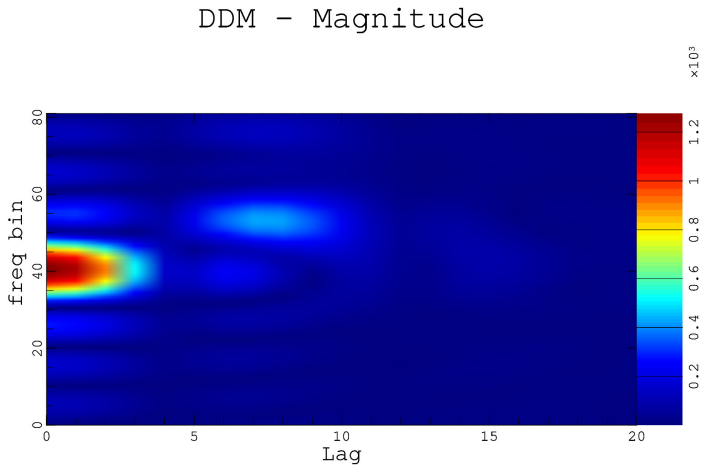
Figure A.7: First DDM (N=8000000 and n_0=20) of Balloon detection without DSI mitigation. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -40 Hz (freq bin 0) to 40 Hz (freq bin 81).



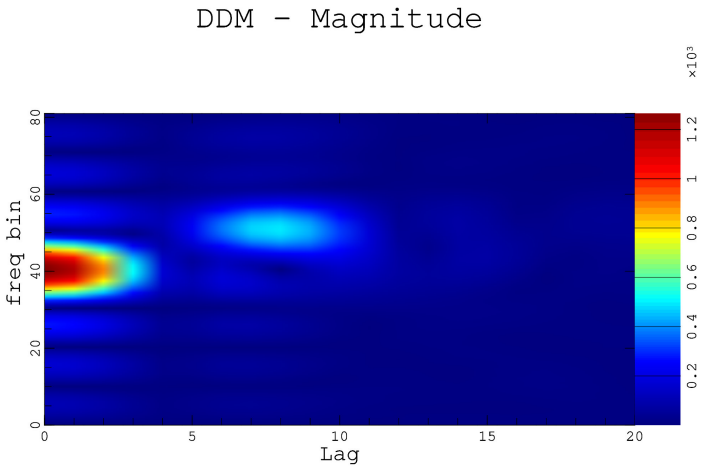
(a) DDM with N=8000000 and n_0=8000020



(b) DDM with N=8000000 and n_0=16000020

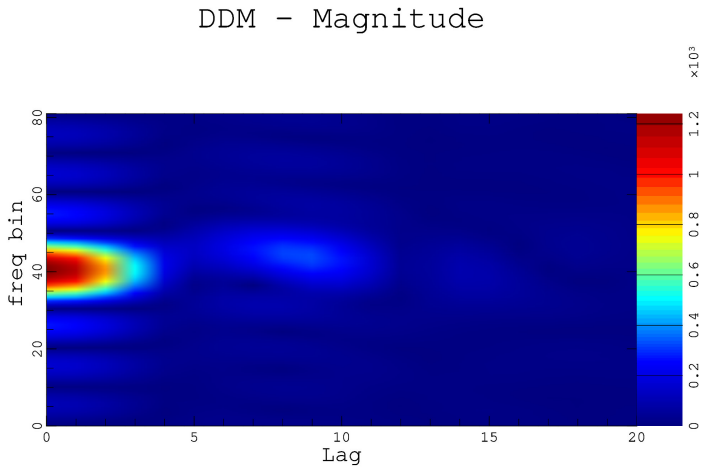


(c) DDM with N=8000000 and n_0=24000020

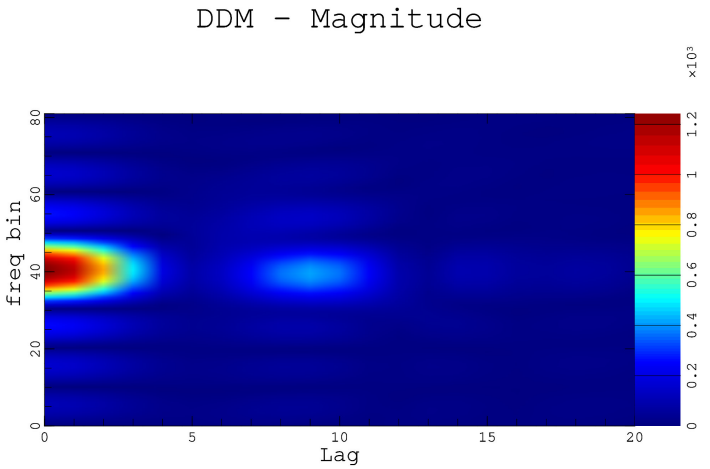


(d) DDM with N=8000000 and n_0=32000020

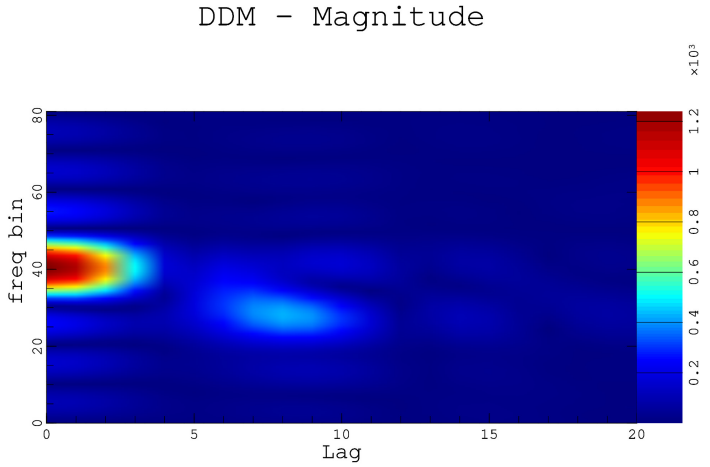
Figure A.8: Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (first part) of Balloon detection without DSI mitigation. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -40 Hz (freq bin 0) to 40 Hz (freq bin 81).



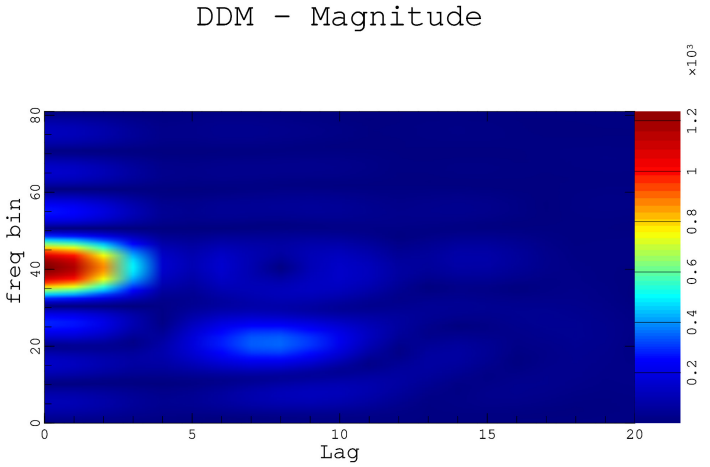
(a) DDM with N=8000000 and n_0=40000020



(b) DDM with N=8000000 and n_0=48000020



(c) DDM with N=8000000 and n_0=56000020

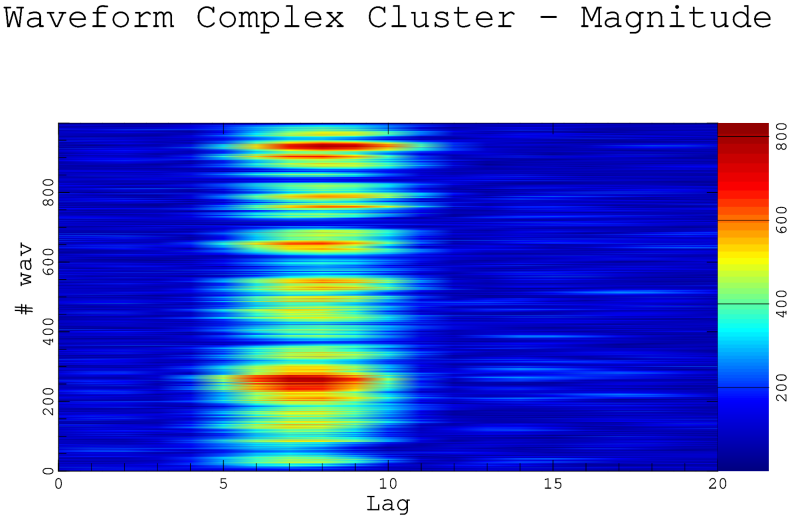


(d) DDM with N=8000000 and n_0=64000020

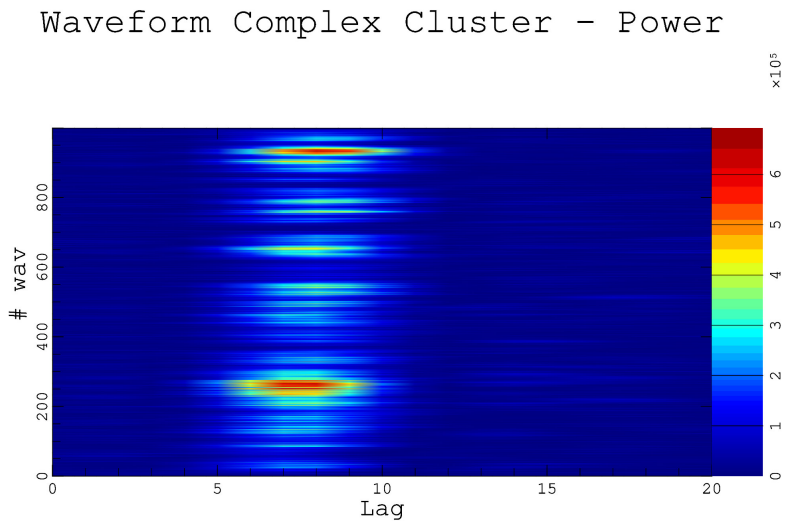
Figure A.9: Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (second part) of Balloon detection without DSI mitigation. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -40 Hz (freq bin 0) to 40 Hz (freq bin 81).

A.2.2. Second before

In this section the graphs obtained, in reference to the detection of the balloon, of the seconds before and after those shown in the report 6.2. are presented.



(a) Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(b) Power of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.

Figure A.10: Second before. Complex cross-correlation waveform clusters of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

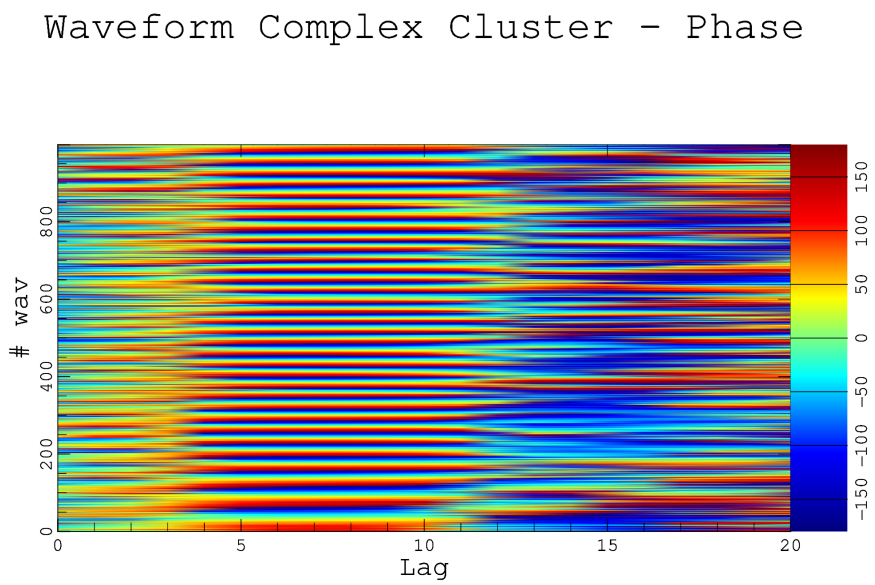


Figure A.11: Second before. Phase of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

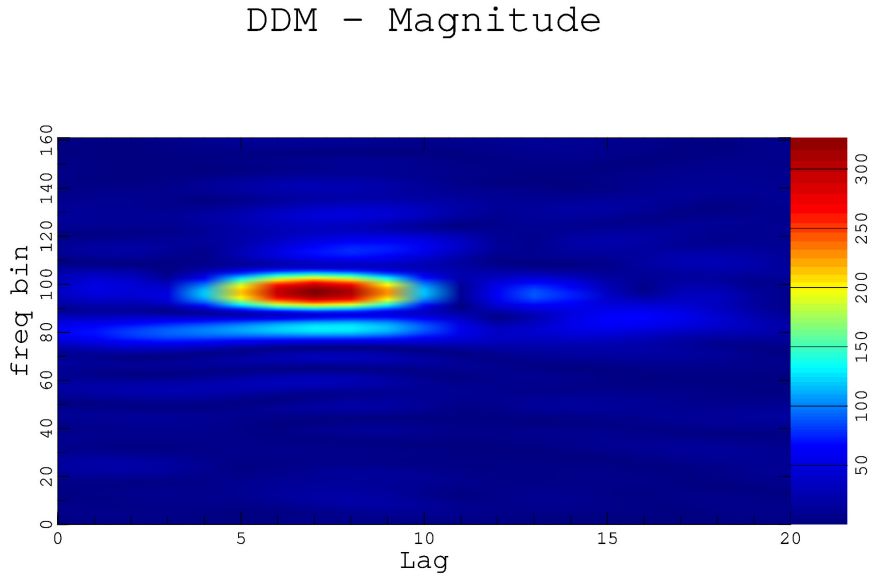
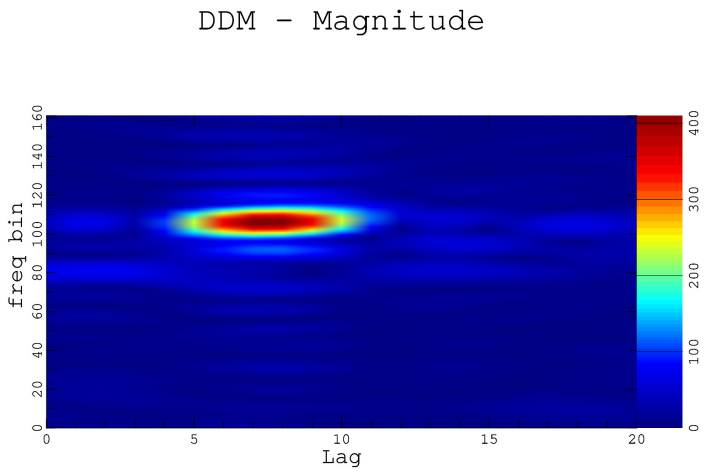
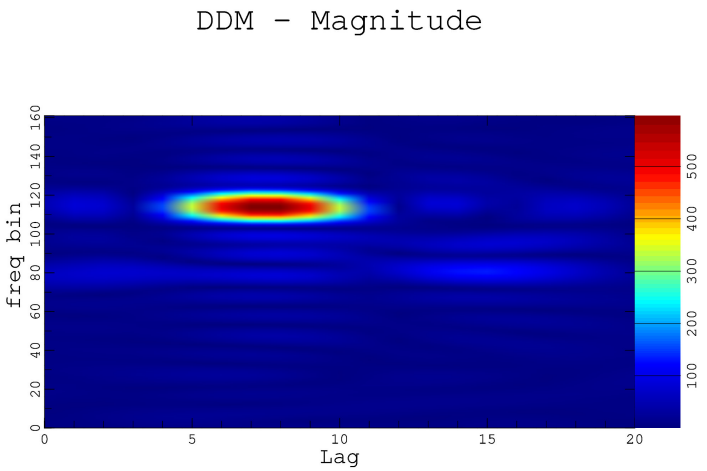


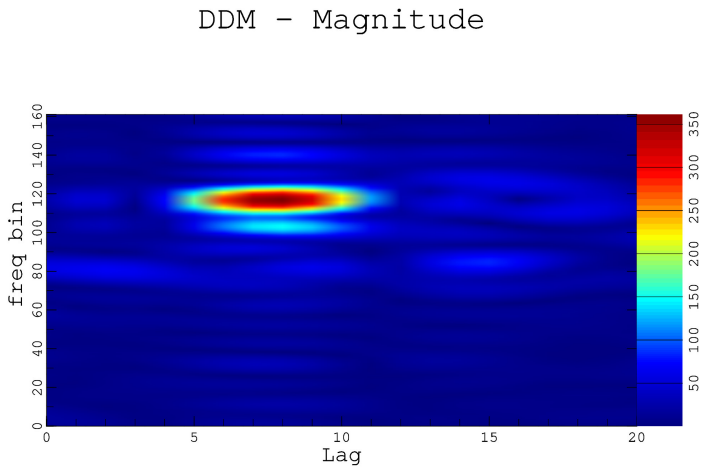
Figure A.12: Second before. First DDM (N=8000000 and n_0=20) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -80 Hz (freq bin 0) to 80 Hz (freq bin 161).



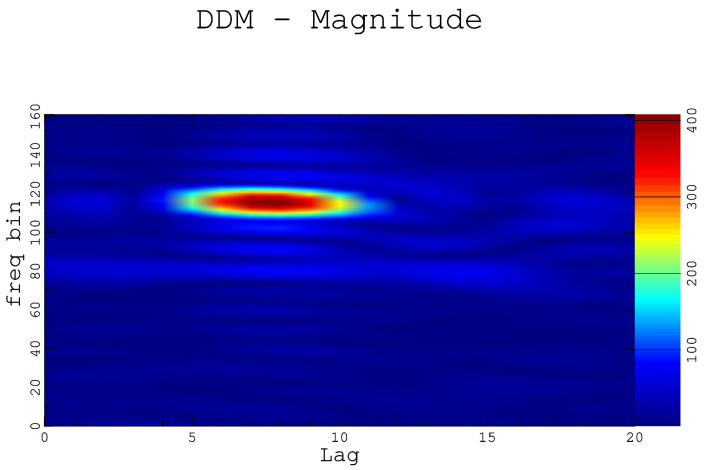
(a) DDM with N=8000000 and n_0=8000020



(b) DDM with N=8000000 and n_0=16000020

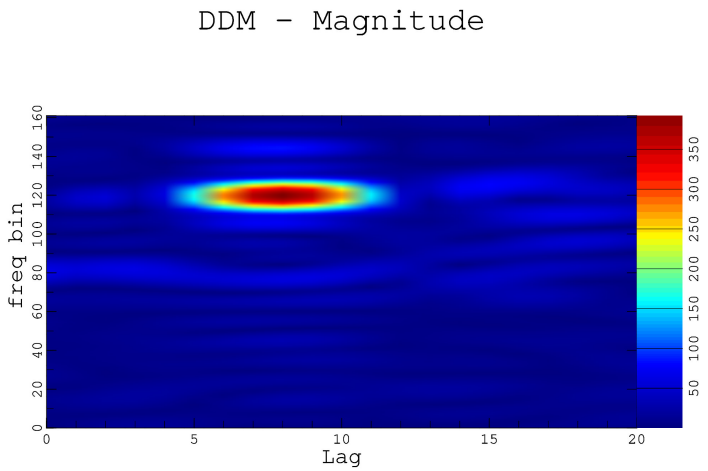


(c) DDM with N=8000000 and n_0=24000020

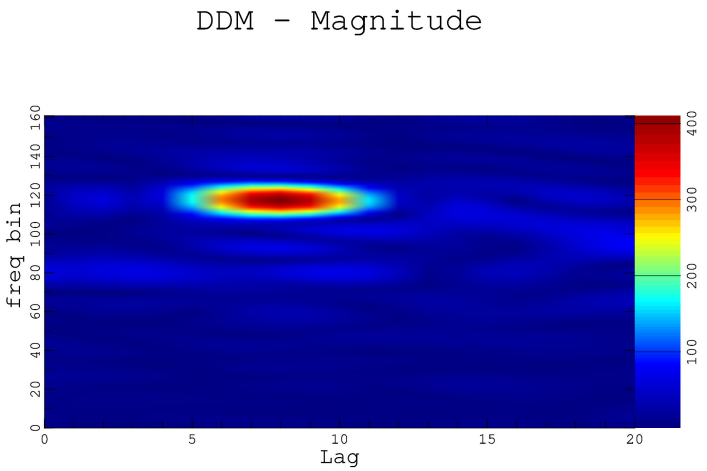


(d) DDM with N=8000000 and n_0=32000020

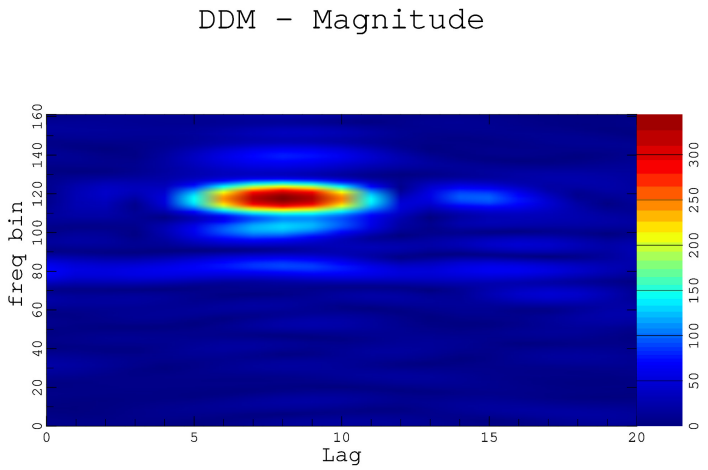
Figure A.13: Second before. Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (first part) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -80 Hz (freq bin 0) to 80 Hz (freq bin 161).



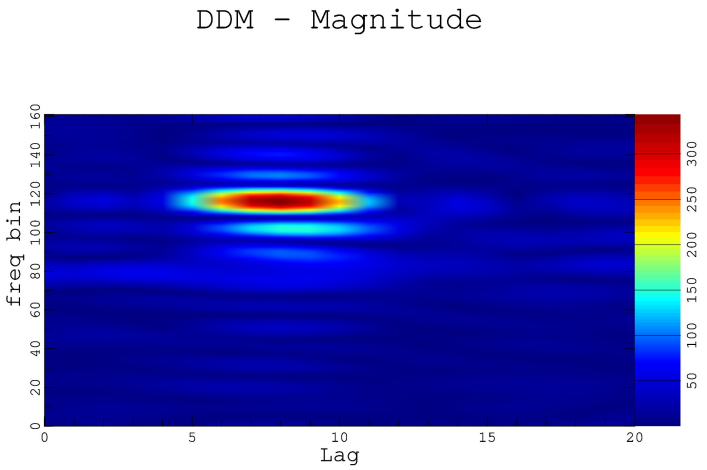
(a) DDM with N=8000000 and n_0=40000020



(b) DDM with N=8000000 and n_0=48000020



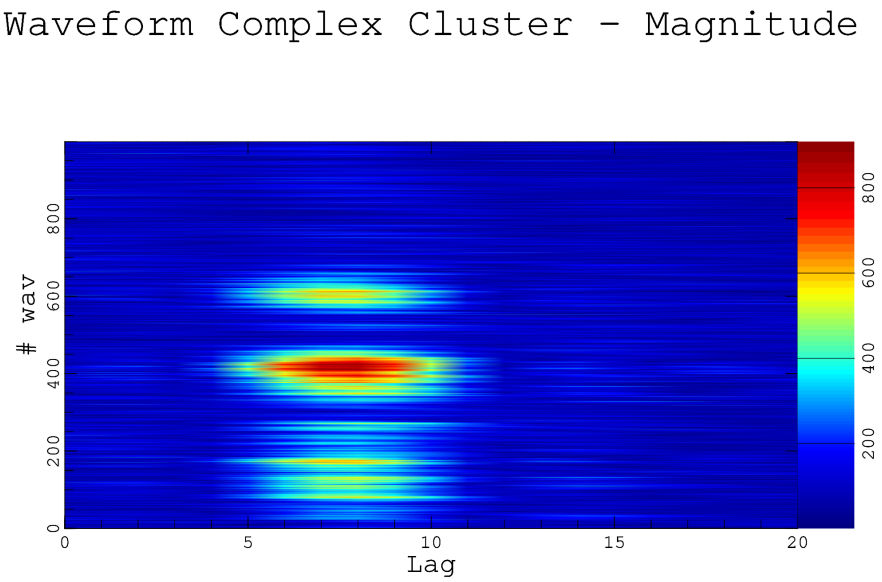
(c) DDM with N=8000000 and n_0=56000020



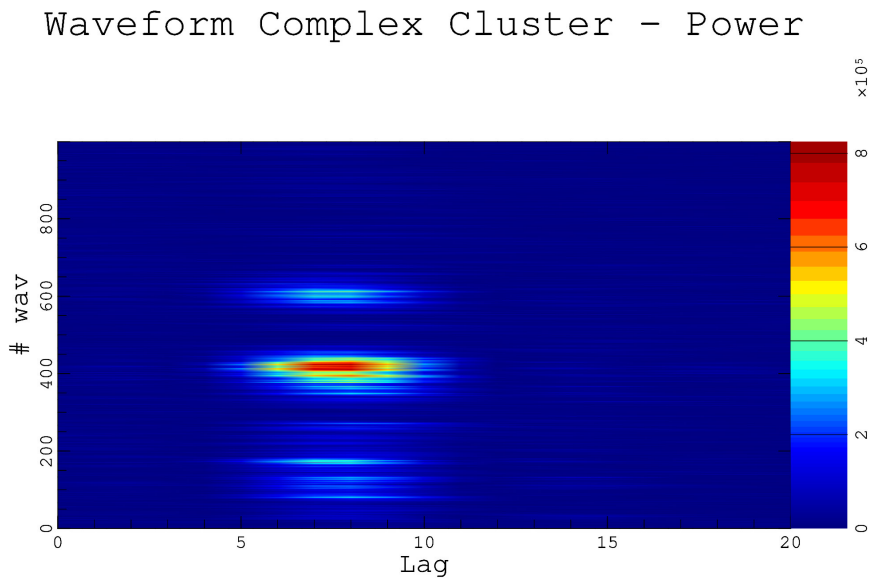
(d) DDM with N=8000000 and n_0=64000020

Figure A.14: Second before. Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (second part) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -80 Hz (freq bin 0) to 80 Hz (freq bin 161).

A.2.3. Second after



(a) Magnitude of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.



(b) Power of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each.

Figure A.15: Second after. Complex cross-correlation waveform clusters of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

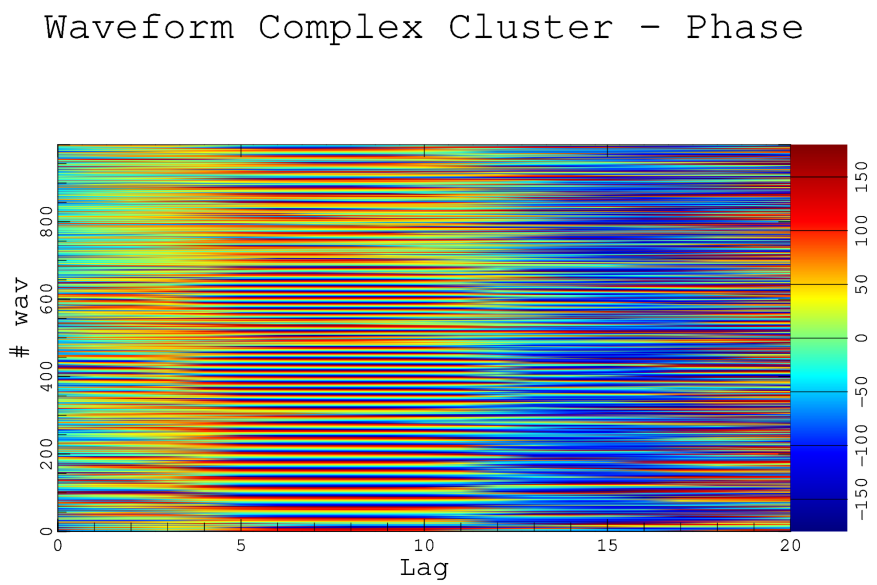


Figure A.16: Second after. Phase of complex cross-correlation waveform cluster of 1 s, composed of complex cross-correlation waveforms of 1 ms each of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz).

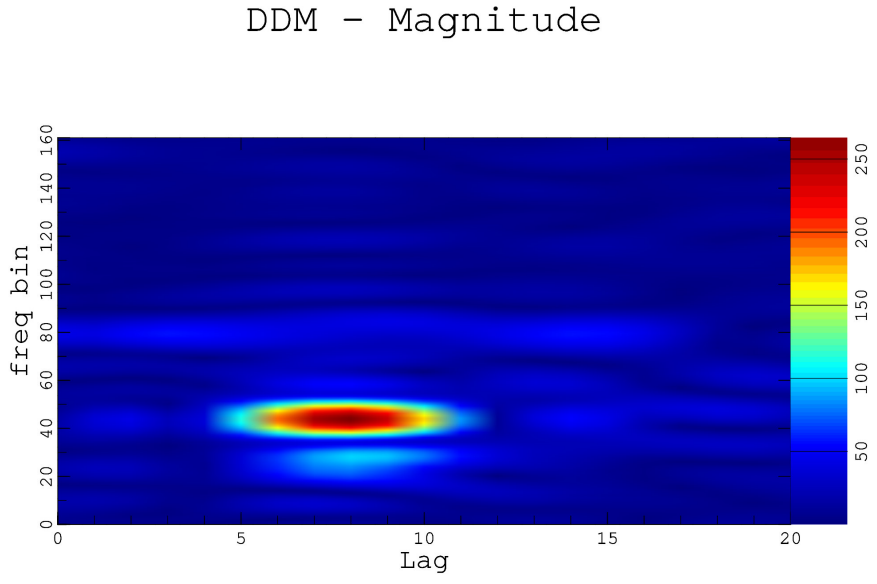
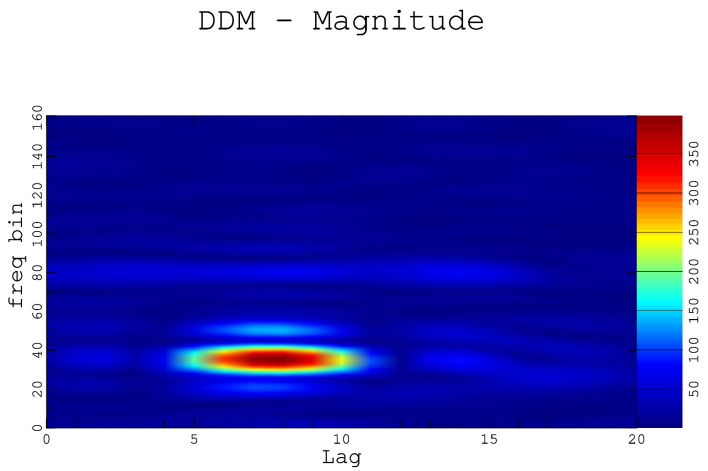
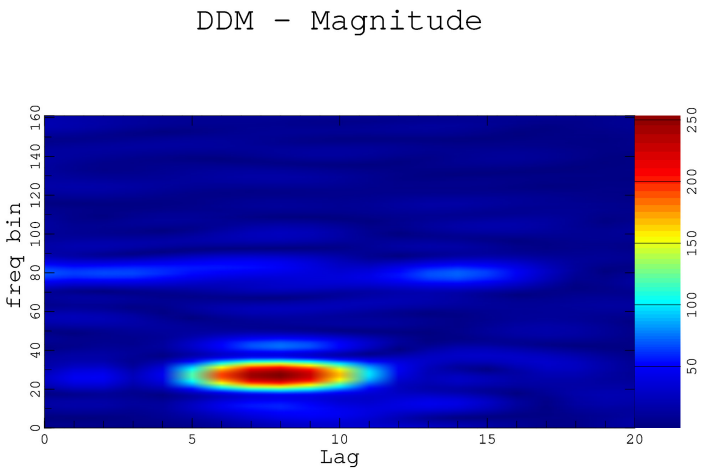


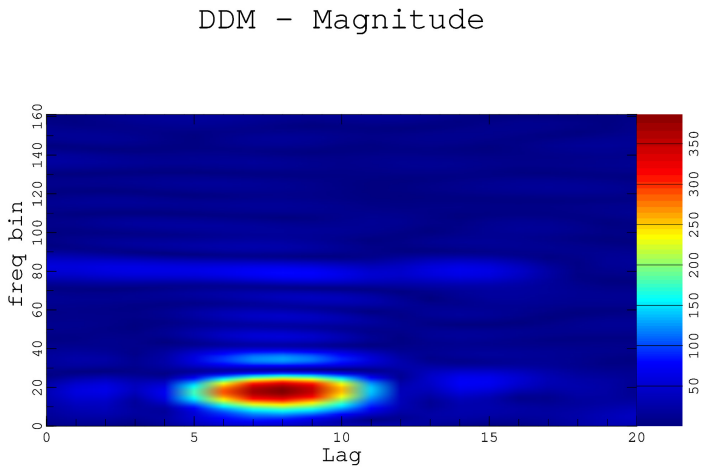
Figure A.17: Second after. First DDM (N=8000000 and n_0=20) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -80 Hz (freq bin 0) to 80 Hz (freq bin 161).



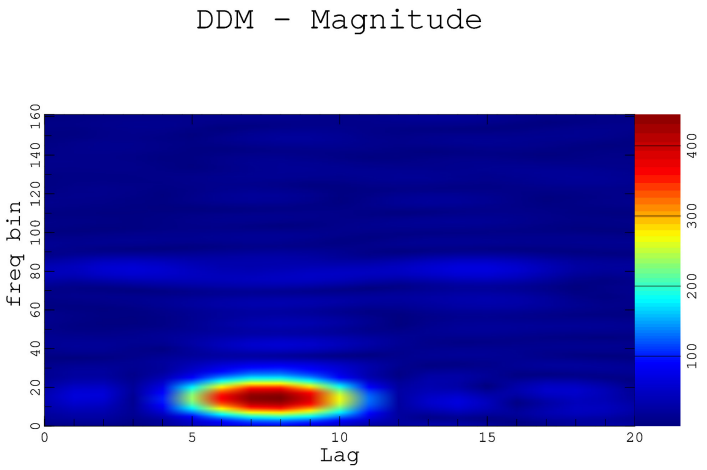
(a) DDM with N=8000000 and n_0=8000020



(b) DDM with N=8000000 and n_0=16000020

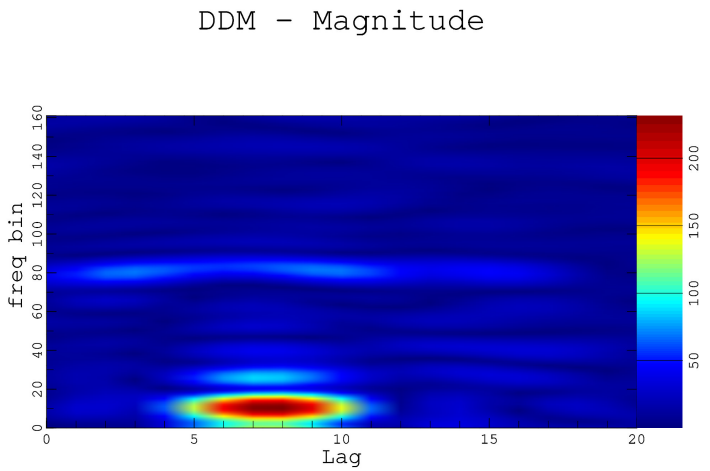


(c) DDM with N=8000000 and n_0=24000020

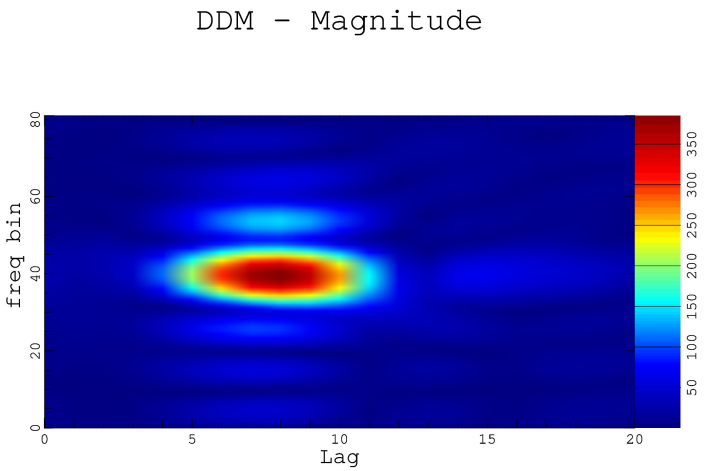


(d) DDM with N=8000000 and n_0=32000020

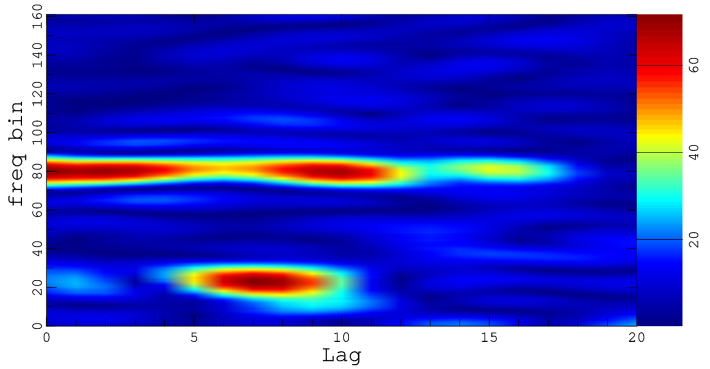
Figure A.18: Second after. Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (first part) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -80 Hz (freq bin 0) to 80 Hz (freq bin 161).



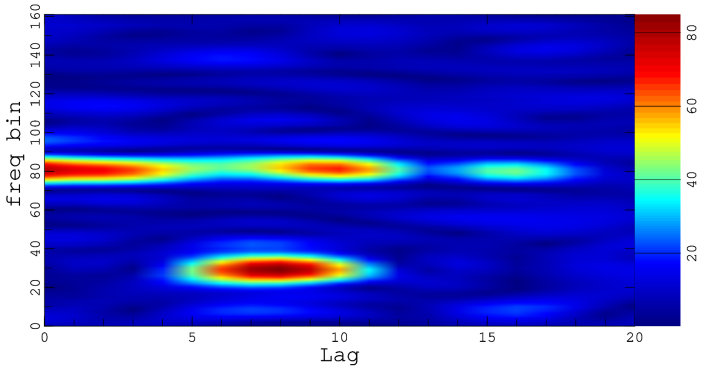
(a) DDM with N=8000000 and n_0=40000020



(b) DDM with N=8000000 and n_0=48000020



(c) DDM with N=8000000 and n_0=56000020



(d) DDM with N=8000000 and n_0=64000020

Figure A.19: Second before. Delay-Doppler maps during 1 second with 1 ms time interval each subfigure graph (second part) of Balloon detection. Using DVB-S/DVB-S2 opportunity signal transmitted by 19.2°E ASTRA 1KR satellite with carrier frequency f_c = 11318 MHz (applying digital filtering with a 7^{th} order Butterworth filter with cutoff frequency f_c = 10 MHz). From -80 Hz (freq bin 0) to 80 Hz (freq bin 161).

APPENDIX B. LNB DATASHEET

Low phase noise and high IP3.

The professional PLL has low phase noise, high IP3 and low noise figure.

Options include separate input for the ext. 10 MHz ref., customized LO, customized gain and separate DC power input connector.

All our LNBs are individually hand tuned to get the very best performance available for each unit. Quality and long term reliability is also essential. Therefore are all LNBs tested according to a very extensive test program, which includes heating, cooling, water-proof testing and rigorous electrical testing.

Swedish Microwave was founded 1986 and is today a leading manufacturer of professional LNBs (Low Noise Block converters). The company is located in Motala Sweden, and to date the products are installed in more than 100 countries.

All work is in-house allowing custom-design products, short delivery times, high flexibility, quick service and support.

SPECIFICATION SMW PLL-LNB ext. 10 MHz ref.

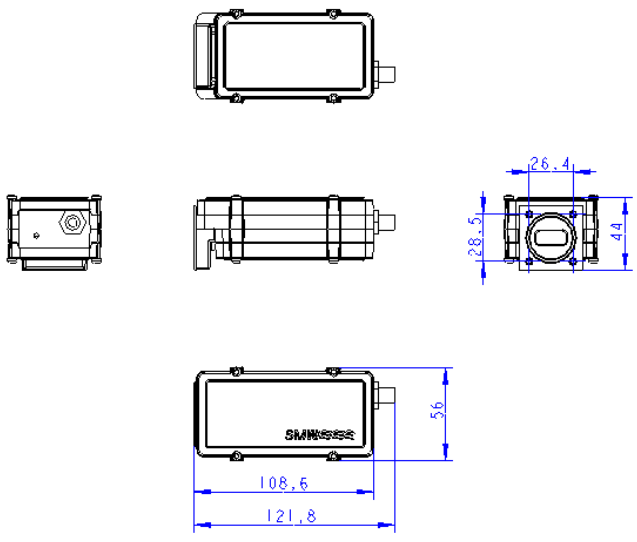
SMW PLL-LNB ext. 10 MHz ref.	***9.75 GHz***	***10.0 GHz***	***10.25 GHz***	***10.5 GHz***
Frequency range	10.7 - 11.8 GHz	10.95 - 12.1 GHz	11.2 - 11.7 GHz	11.45 - 12.2 GHz
LO frequency	9.75 GHz	10.0 GHz	10.25 GHz	10.5 GHz
Output frequency	950 - 2050 MHz	950 - 2100 MHz	950 - 1450 MHz	950 - 1700 MHz

SMW PLL-LNB ext. 10 MHz ref.	***10.6 GHz***	***10.7 GHz***	***10.75 GHz***	***11.2 GHz***
Frequency range	11.7 - 12.75 GHz	11.65 - 12.75 GHz	11.7 - 12.75 GHz	12.2 - 12.75 GHz
LO frequency	10.6 GHz	10.7 GHz	10.75 GHz	11.2 GHz
Output frequency	1100 - 2150 MHz	950 - 2050 MHz	950 - 2000 MHz	1000 - 1550 MHz

SMW PLL-LNB ext. 10 MHz ref.	***11.25 GHz***	***11.3 GHz***	***11.475 GHz***
Frequency range	12.2 - 12.75 GHz	12.25 - 12.75 GHz	12.5 - 12.75 GHz
LO frequency	11.25 GHz	11.3 GHz	11.475 GHz
Output frequency	950 - 1500 MHz	950 - 1450 MHz	1025 - 1275 MHz

LO stability (over temp.)*	depend on the external reference
External reference input frequency	10 MHz
External reference input power	-5 to +10 dBm
Gain	60 dB typ. (55 dB min.)
Gain variation within 30 MHz max.	±0.4 dB
Gain variation max.	±4 dB
Noise Figure, typical	0.8 dB
LO Phase noise typical	-70 dBc @ 10 Hz -70 dBc @ 100 Hz -75 dBc @ 1 kHz -85 dBc @ 10 kHz -100 dBc @ 100 kHz
External Reference Phase noise	-130 dBc @ 10 Hz -135 dBc @ 100 Hz -143 dBc @ 1 kHz -145 dBc @ 10 kHz -160 dBc @ 100 kHz
LO radiation	-60 dBm
Image rejection	40 dB min
P1dB typ.	+15 dBm
IP 3 typ.	+25 dBm
Input	WR-75 waveguide (R120)
External reference input port	Output IF connector. Option: Sep. connector (F, N or SMA)
Output (waterproof)	F-connector 75 ohm, N-connector 50 ohm or SMA-connector 50 ohm
Input VSWR	2.3:1 max
Output VSWR	2.1:1 max
DC power	12 - 24 V 270 mA typ.
Operating temperature	-40 to +80 ℃
Storage temperature	-40 to +80 ℃
Dimensions	122 (127 N) x 56 x 44 mm
Weight	329 g (F- & SMA-connector) 345 g (N-connector)

Options	10 MHz ref. via separate input connector (F, N or SMA) Separate DC power input (F, N or SMA) Customized gain and variation Customized LO SMA-input (via transition) Extended frequency range
Enclosed accessories	O-ring Screw M4 x 8 4 pcs



Specifications are subject to change without notice. General terms Orgalime S 2000. Products from Swedish Microwave AB are made for commercial use. 2010-08-24

SMW SWEDISH MICROWAVE AB
Dynamovägen 5 | 591 61 Motala | SWEDEN | Tel. +46 141 216135 | Fax. +46 141 215224 | info@smw.se | www.smw.se

APPENDIX C. CAR TRAJECTORY DATA

CSRS-PPP (V 1.05 11216)

CAROK1222k_QWHS

Data Start	Data End	Duration of Observations
2017-12-22 10:00:00.000	2017-12-22 10:34:01.000	0h 34m 1.00s

Apri / Aposteriori Code Std

2.0m / 1.496m

Observations	Frequency	Mode
Code	L1	Kinematic

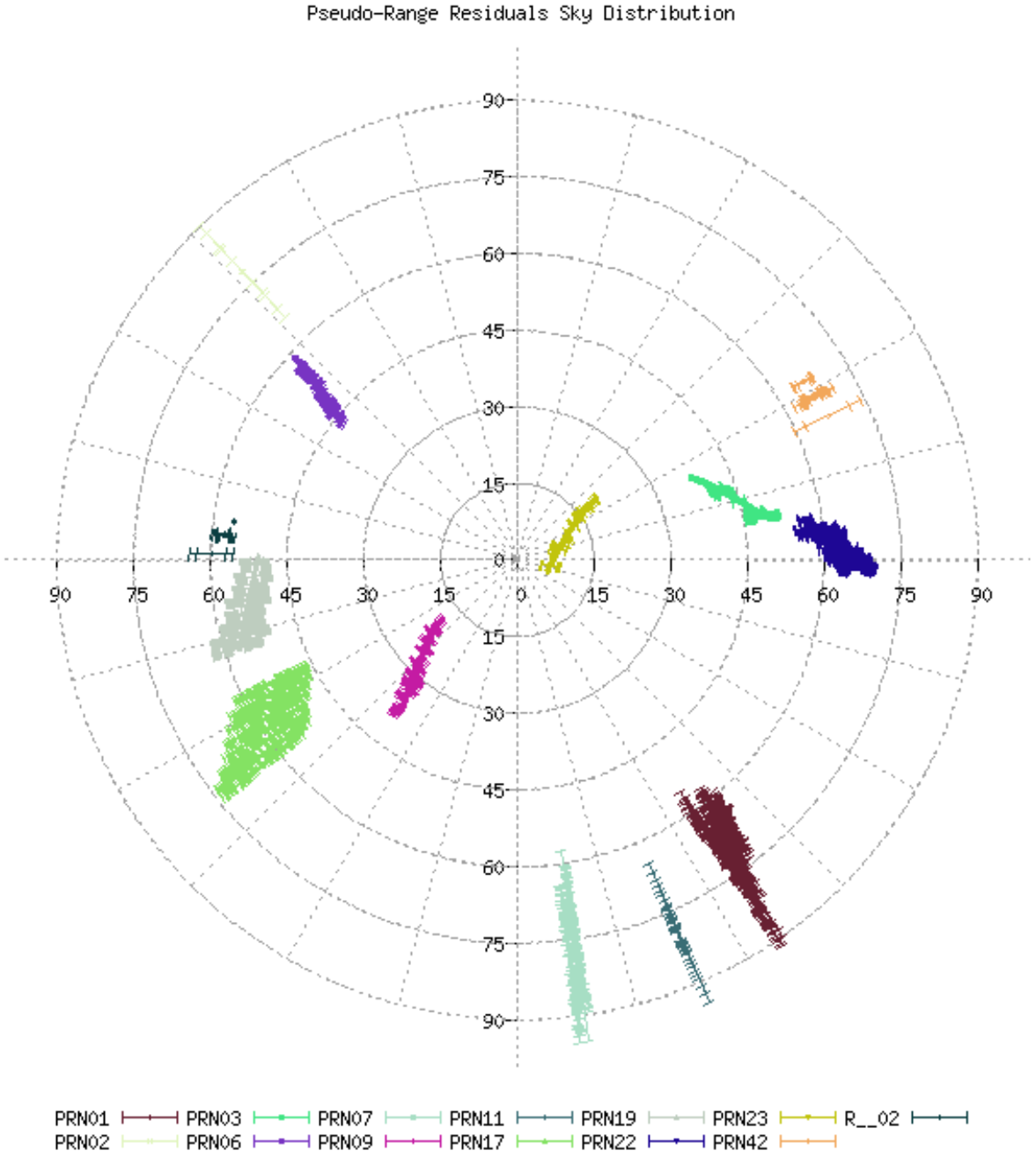
Elevation Cut-Off	Rejected Epochs	Observation & Estimation Steps
10.000 degrees	0.73 %	1.00 sec / 1.00 sec

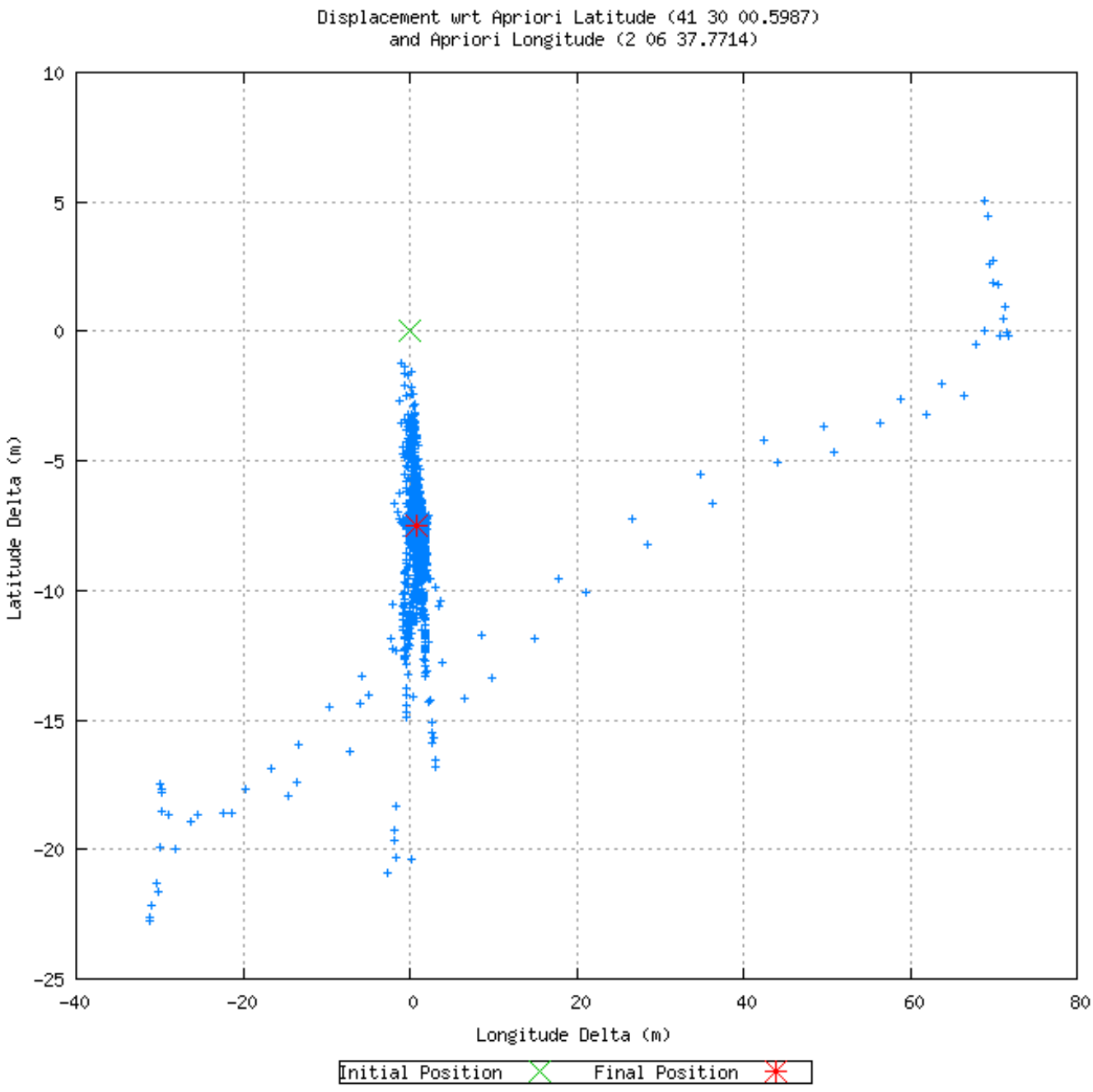
Antenna Model	APC to ARP	ARP to Marker
-Unknown-	Ant. not in PPP (0 m)	0.000 m

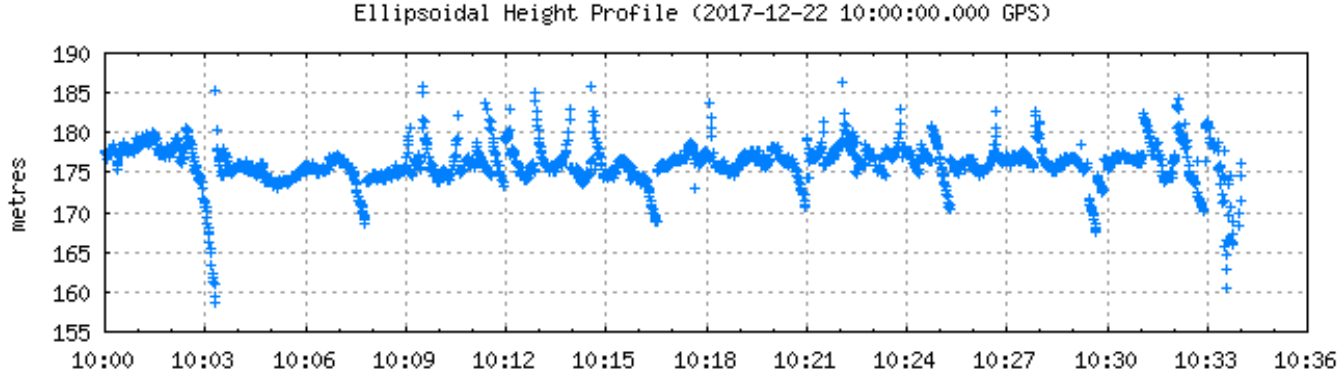
(APC = antenna phase center; ARP = antenna reference point)

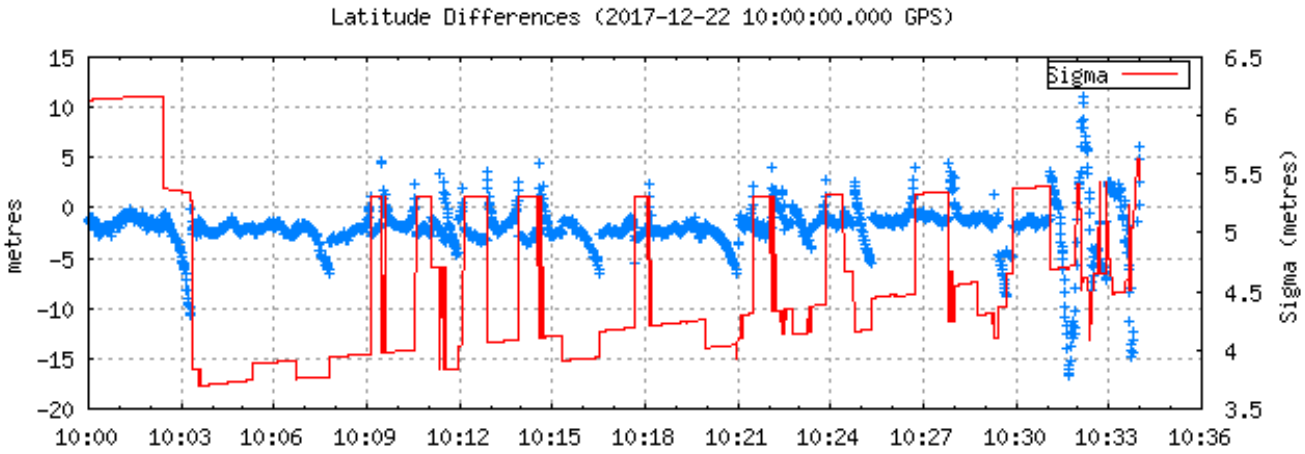
Estimated NAD83 Positions for CARO356k.17O can be found in CARO356k.17O.pos file

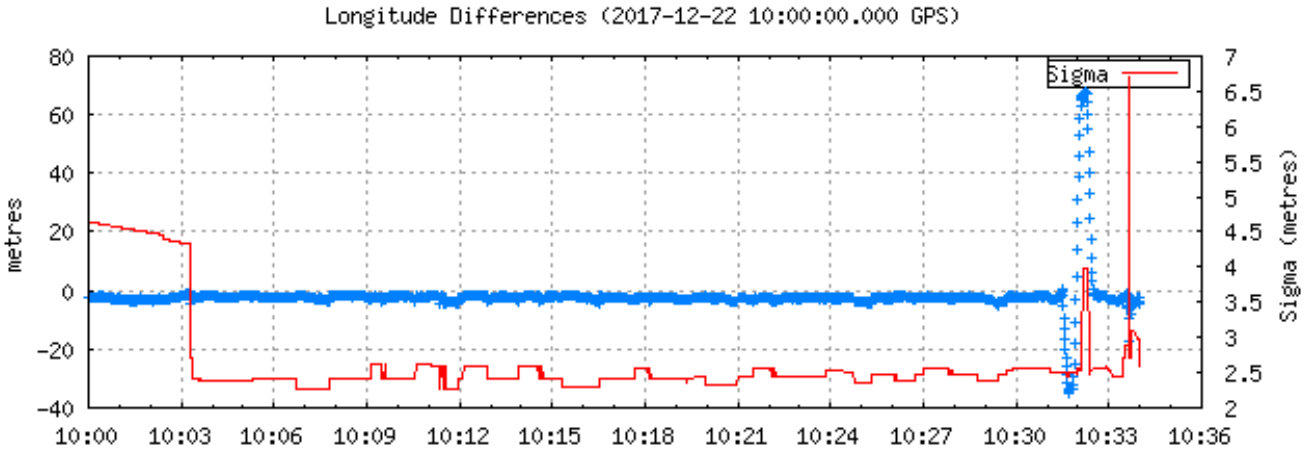
Estimated Parameters & Observations Statistics

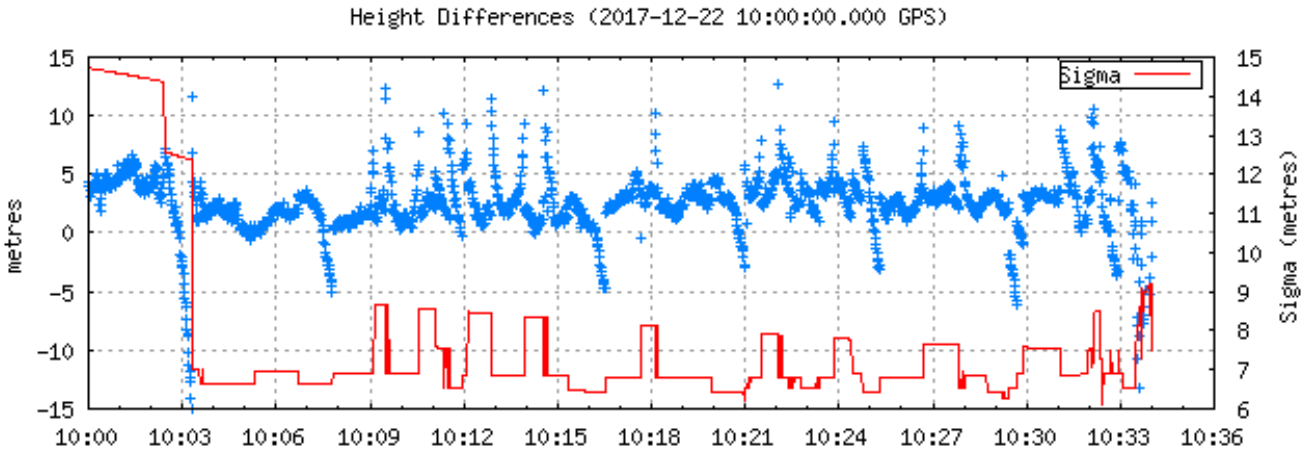


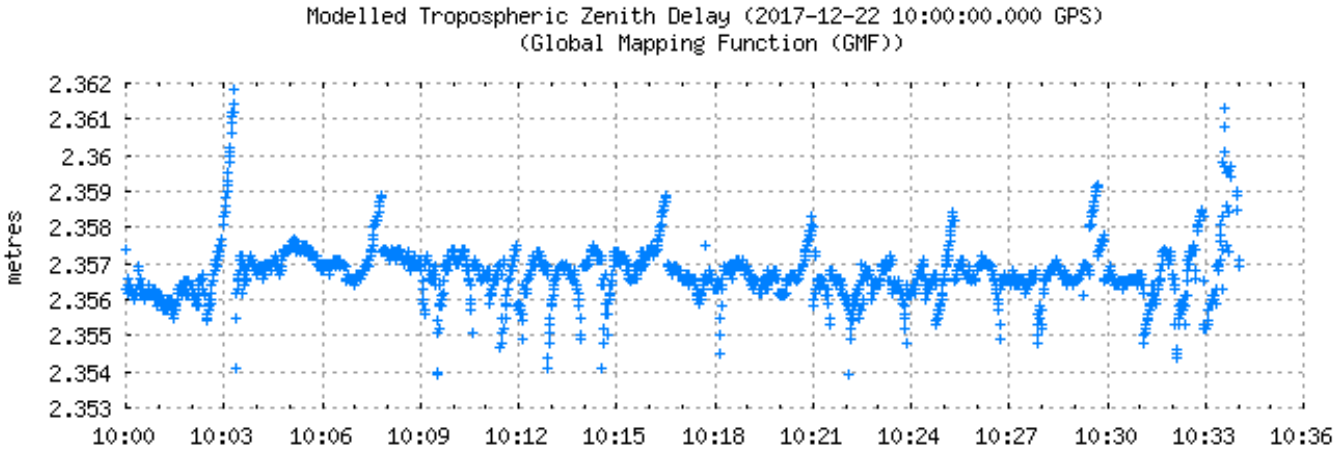

Displacement wrt Apriori Latitude (41 30 00.5987)
and Apriori Longitude (2 06 37.7714)
Latitude Delta (m)
Longitude Delta (m)
10
5
0
-5
-10
-15
-20
-25
-40
-20
0
20
40
60
80
Initial Position
Final Position

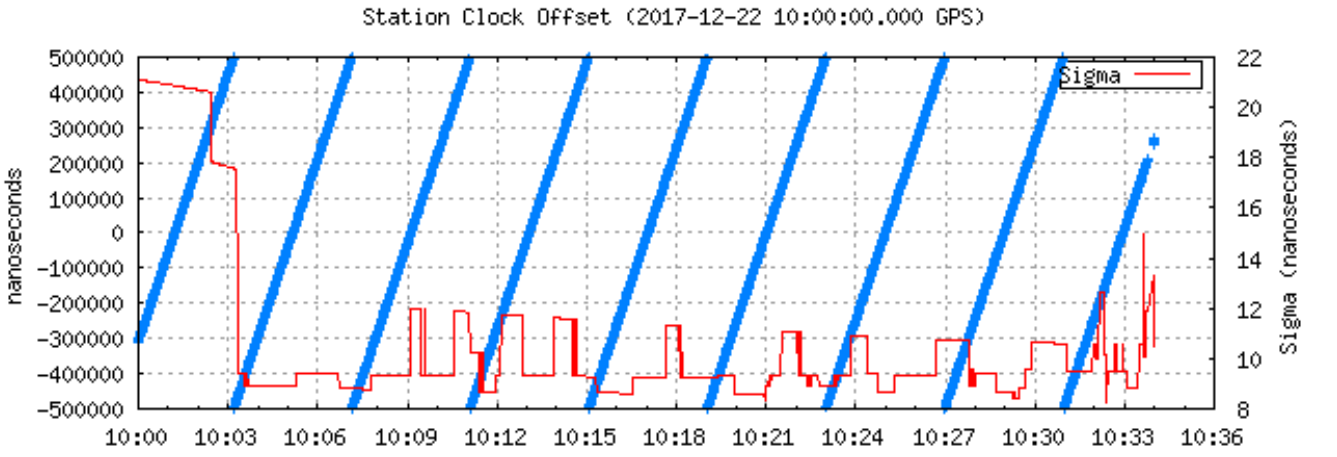

Ellipsoidal Height Profile (2017-12-22 10:00:00.000 GPS)
metres
190
185
180
175
170
165
160
155
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36

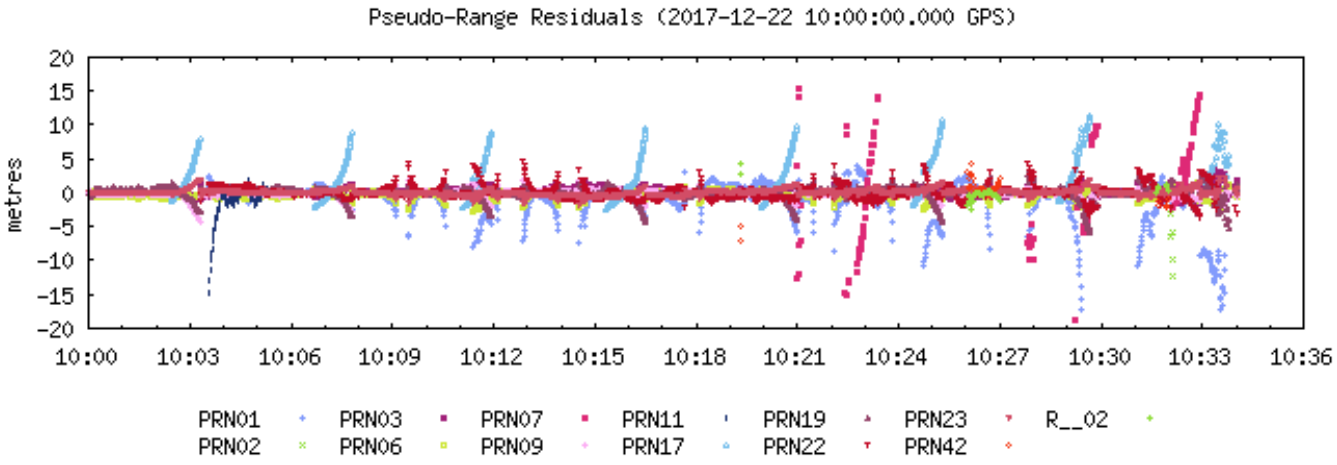

Latitude Differences (2017-12-22 10:00:00.000 GPS)
Sigma
metres
Sigma (metres)
15
10
5
0
-5
-10
-15
-20
6.5
6
5.5
5
4.5
4
3.5
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36


Longitude Differences (2017-12-22 10:00:00.000 GPS)
Sigma
metres
Sigma (metres)
80
60
40
20
0
-20
-40
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36


Height Differences (2017-12-22 10:00:00.000 GPS)
Sigma
metres
Sigma (metres)
15
10
5
0
-5
-10
-15
14
13
12
11
10
9
8
7
6
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36


Modelled Tropospheric Zenith Delay (2017-12-22 10:00:00.000 GPS)
(Global Mapping Function (GMF))
metres
2.362
2.361
2.36
2.359
2.358
2.357
2.356
2.355
2.354
2.353
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36


Station Clock Offset (2017-12-22 10:00:00.000 GPS)
nanoseconds
Sigma (nanoseconds)
Sigma
500000
400000
300000
200000
100000
0
-100000
-200000
-300000
-400000
-500000
22
20
18
16
14
12
10
8
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36


Pseudo-Range Residuals (2017-12-22 10:00:00.000 GPS)
metres
20
15
10
5
0
-5
-10
-15
-20
10:00
10:03
10:06
10:09
10:12
10:15
10:18
10:21
10:24
10:27
10:30
10:33
10:36
PRN01
PRN02
PRN03
PRN06
PRN07
PRN09
PRN11
PRN17
PRN19
PRN22
PRN23
PRN42
R__02

~~~ Disclaimer ~~~

Natural Resources Canada does not assume any liability deemed to have been caused directly or indirectly by any content of its PPP-On-Line positioning service.

If you have any questions, please feel free to contact:
EMail: nrcan.geodeticinformationservices.rncan@canada.ca
Phone:343-292-6617

Natural Resources Canada | Ressources naturelles Canada

Canada

APPENDIX D. CODES

D.1. C++

D.1.1. Digital filtering

D.1.1.1. filter.h

```
/**
  \file filter.h
  \brief Public header file for the class Filter
*/

#ifndef _FILTER_H_
#define _FILTER_H_

#include <string>

class Filter
{
  // Member variables
  private:
    unsigned int m_Nfir;
    unsigned int m_Niir;
    double* m_a = NULL;
    double* m_b = NULL;
    double* m_tabsreal = NULL;
    double* m_tabsimag = NULL;


  public:
    // Constructors & destructor
    Filter(unsigned int Nfir, unsigned int Niir, double* b, double* a,
        double* tabsreal, double* tabsimag);
    ~Filter();

   // Set things
   void SetFIR(unsigned int Nfir, double* b);
   void SetIIR(unsigned int Niir, double* a);
   void SetTabs(unsigned int L, double* tabsreal, double* tabsimag);
   void ClearTabs();

   // Get things
   unsigned int GetNfir();
   unsigned int GetNiir();
   double* Geta();
   double* Getb();
   double* GetTabsReal();
   double* GetTabsImag();
```

```
41
42    // Print
43    void Print();
44
45    // Read from files
46    static Filter* GetFromFilterFile(std::string filename);
47
48   private:
49     void FreeMemory();
50 };
51
52
53 #endif // _FILTER_H_
```

D.1.1.2. filter.cc

```
1  /**
2    \file filter.cc
3    \brief Contains the definitions for class Filter
4  */
5
6  #ifndef _FILTER_CC_
7  #define _FILTER_CC_
8
9  #include "filter.h"
10
11 #include <string.h>
12 #include <stdlib.h>
13 #include <iostream>
14 #include <fstream>
15
16 #include <string>
17 #include <sstream>
18 #include <vector>
19 #include <iterator>
20
21 //-----------------------------------------------------------
22 Filter::Filter(unsigned int Nfir, unsigned int Niir, double* b, double*
       a, double* tabsreal, double* tabsimag)
23 {
24   m_Nfir = Nfir;
25   m_Niir = Niir;
26
27   // Allocate memory for m_a
28   m_a = (double*) calloc(Niir+1, sizeof(double));
29   if (NULL==m_a)
30   {
31     std::cerr << "ERROR: Could not allocate memory for m_a." << std::endl;
32     exit(EXIT_FAILURE);
```

```
33    }
34    // Copy data from a into m_a
35    memcpy(m_a, a, (Niir+1)*sizeof(double));
36
37    // Allocate memory for m_b
38    m_b = (double*) calloc(Nfir+1, sizeof(double));
39    if (NULL==m_b)
40    {
41      std::cerr << "ERROR: Could not allocate memory for m_b." << std::endl;
42      exit(EXIT_FAILURE);
43    }
44    // Copy data from b into m_b
45    memcpy(m_b, b, (Nfir+1)*sizeof(double));
46
47    // Allocate memory for m_tabsreal
48    int maxN = Niir;
49    if(Nfir>Niir)
50        maxN = Nfir;
51
52    m_tabsreal = (double*) calloc(maxN+1, sizeof(double));
53    if (NULL==m_tabsreal)
54    {
55      std::cerr << "ERROR: Could not allocate memory for m_tabsreal." <<
          std::endl;
56      exit(EXIT_FAILURE);
57    }
58    // Copy data from tabs into m_tabsreal
59    if (Nfir>=Niir)
60    {
61      memcpy(m_tabsreal, tabsreal, (Nfir+1)*sizeof(double));
62    }
63    if (Nfir<Niir)
64    {
65      memcpy(m_tabsreal, tabsreal, (Niir+1)*sizeof(double));
66    }
67
68    // Allocate memory for m_tabsimag
69    m_tabsimag = (double*) calloc(maxN+1, sizeof(double));
70    if (NULL==m_tabsimag)
71    {
72      std::cerr << "ERROR: Could not allocate memory for m_tabsimag." <<
          std::endl;
73      exit(EXIT_FAILURE);
74    }
75    // Copy data from tabs into m_tabsimag
76    if (Nfir>=Niir)
77    {
78      memcpy(m_tabsimag, tabsimag, (Nfir+1)*sizeof(double));
79    }
80    if (Nfir<Niir)
81    {
```

```
82      memcpy(m_tabsimag, tabsimag, (Niir+1)*sizeof(double));
83    }
84
85
86    /*
87    SetFIR(Nfir, b);
88    SetIIR(Niir, a);
89    SetTabs(tabs);
90    */
91  }
92
93  //-------------------------------------------------------
94  Filter::~Filter()
95  {
96    /** Frees memory */
97    FreeMemory();
98  }
99
100 //-------------------------------------------------------
101 void Filter::SetFIR(unsigned int Nfir, double* b)
102 {
103   m_Nfir = Nfir;
104   m_b = b;
105 }
106
107 //-------------------------------------------------------
108 void Filter::SetIIR(unsigned int Niir, double* a)
109 {
110   m_Niir = Niir;
111   m_a = a;
112 }
113
114 //-------------------------------------------------------
115 void Filter::SetTabs(unsigned int L, double* tabsreal, double* tabsimag)
116 {
117   int maxN = m_Niir;
118   if(m_Nfir>m_Niir)
119   {
120     maxN = m_Nfir;
121   }
122   if (L!=maxN+1)
123   {
124       std::cerr << "ERROR: cannot assign tab content to filter with
          different length. " << std::endl;
125       exit(EXIT_FAILURE);
126   }
127   memcpy(m_tabsreal, tabsreal, L*sizeof(double));
128   memcpy(m_tabsimag, tabsimag, L*sizeof(double));
129 }
130
131 //-------------------------------------------------------
```

```
132 void Filter::ClearTabs()
133 {
134   int maxN = m_Niir;
135   if (m_Nfir >m_Niir)
136   {
137     maxN = m_Nfir;
138   }
139   for (unsigned int k=0; k<=maxN; k++)
140   {
141     m_tabsreal[k] = 0;
142     m_tabsimag[k] = 0;
143   }
144 }
145 //---------------------------------------------------------
146 unsigned int Filter::GetNfir()
147 {
148     return m_Nfir;
149 }
150
151 //---------------------------------------------------------
152 unsigned int Filter::GetNiir()
153 {
154     return m_Niir;
155 }
156
157 //---------------------------------------------------------
158
159 double* Filter::Geta()
160 {
161     return m_a;
162 }
163
164 //---------------------------------------------------------
165
166 double* Filter::Getb()
167 {
168     return m_b;
169 }
170
171 //---------------------------------------------------------
172
173 double* Filter::GetTabsReal()
174 {
175     return m_tabsreal;
176 }
177
178 double* Filter::GetTabsImag()
179 {
180     return m_tabsimag;
181 }
182
```

```
183 //-------------------------------------------------------
184
185
186 Filter* Filter::GetFromFilterFile(std::string filename)
187 {
188
189     unsigned int Nfir;
190     unsigned int Niir;
191
192     double* a = NULL;;
193     double* b = NULL;;
194     double* tabsreal = NULL;;
195     double* tabsimag = NULL;
196
197     std::ifstream file(filename.c_str());
198     std::string line;
199     unsigned int cont=1;
200     unsigned int k=0;
201
202     while(getline(file, line))
203     {
204         if (cont==1 or cont==2)
205         {
206             std::stringstream s_str(line);
207             if (cont ==1)
208                 s_str>>Nfir;
209             else
210             {
211                 s_str>>Niir;
212             }
213         }
214         else
215         {
216             std::istringstream buf(line);
217             std::istream_iterator<std::string> beg(buf), end;
218             std::vector<std::string> tokens(beg, end); // done!
219
220             // Allocate memory
221             if (k==0)
222             {
223                 // a
224                 a = (double*) calloc(Niir+1, sizeof(double));
225                 if (NULL==a)
226                 {
227                     std::cerr << "ERROR: Could not allocate memory for a." <<
                            std::endl;
228                     exit(EXIT_FAILURE);
229                 }
230
231                 // b
232                 b = (double*) calloc(Nfir+1, sizeof(double));
```

```
233             if (NULL==b)
234             {
235                 std::cerr << "ERROR: Could not allocate memory for b." <<
                        std::endl;
236                 exit(EXIT_FAILURE);
237             }
238 
239             // tabsreal
240             if (Nfir>Niir)
241                 tabsreal = (double*) calloc(Nfir+1, sizeof(double));
242             else
243                 tabsreal = (double*) calloc(Niir+1, sizeof(double));
244 
245             if (NULL==tabsreal)
246             {
247                 std::cerr << "ERROR: Could not allocate memory for
                        tabsreal." << std::endl;
248                 exit(EXIT_FAILURE);
249             }
250 
251             // tabsimag
252             if (Nfir>Niir)
253                 tabsimag = (double*) calloc(Nfir+1, sizeof(double));
254             else
255                 tabsimag = (double*) calloc(Niir+1, sizeof(double));
256 
257             if (NULL==tabsimag)
258             {
259                 std::cerr << "ERROR: Could not allocate memory for
                        tabsimag." << std::endl;
260                 exit(EXIT_FAILURE);
261             }
262 
263             k = 1;
264         }
265 
266         int i = 0;
267         for(std::string& s: tokens)
268         {
269             if (cont==3)
270             {
271                 std::stringstream s_str(s);
272                 s_str>>b[i];
273             }
274             if (cont==4)
275             {
276                 std::stringstream s_str(s);
277                 s_str>>a[i];
278 
279             }
280             if (cont==5)
```

```
281                 {
282                     std::stringstream s_str(s);
283                     s_str>>tabsreal[i];
284 
285                 }
286                 if (cont==6)
287                 {
288                     std::stringstream s_str(s);
289                     s_str>>tabsimag[i];
290 
291                 }
292                 i++;
293             }
294 
295         }
296         cont++;
297     }
298 
299     Filter* filt = new Filter(Nfir, Niir, b, a, tabsreal, tabsimag);
300 
301 /*    free(a);
302     free(b);
303     free(tabsreal);
304     free(tabsimag);*/
305 
306     return filt;
307 }
308 
309 void Filter::Print()
310 {
311   std::cout << "Nfir = " << GetNfir() << std::endl;
312   std::cout << "Niir = " << GetNiir() << std::endl;
313 
314   printf("b =");
315   double* bi = Getb();
316   for (int i=0; i<GetNfir()+1; i++)
317       printf(" %3.2f",bi[i]);
318 
319   printf("\n");
320   printf("a =");
321   double* ai = Geta();
322   for (int i=0; i<GetNiir()+1; i++)
323       printf(" %3.2f",ai[i]);
324 
325   printf("\n");
326   printf("taps =");
327   double* tapsreal = NULL;
328   double* tapsimag = NULL;
329 
330   tapsreal = GetTabsReal();
331   tapsimag = GetTabsImag();
```

```
332
333   if (GetNfir()>GetNiir())
334   {
335     for (int i=0; i<=GetNfir(); i++)
336         printf(" (%3.2f)i + (%3.2f)j",tapsreal[i], tapsimag[i]);
337   }
338   else
339   {
340     for (int i=0; i<=GetNiir(); i++)
341         printf(" (%3.2f)i + (%3.2f)j\n",tapsreal[i], tapsimag[i]);
342   }
343
344
345 }
346
347 //-------------------------------------------------------------
348 void Filter::FreeMemory()
349 {
350
351   free(m_a);
352   m_a = NULL;
353   free(m_b);
354   m_b = NULL;
355   free(m_tabsreal);
356   m_tabsreal = NULL;
357   free(m_tabsimag);
358   m_tabsimag = NULL;
359 }
360
361
362 #endif // _FILTER_CC_
```

D.1.1.3. Functions Filtering

```
1
2       Apply digital filtering (FIR and IIR) to a signal and create a NEW
            filtered signal.
3
4       @param filt         Pointer of the filter.
5       @return             Returns a pointer to the complex signal.
6     */
7     ComplexSignal* Filtering(Filter* filt);
8
9
10    /**
11      Apply digital filtering (FIR and IIR) to a signal.
12
13      @param filt         Pointer of the filter.
14      @return             The function does not return any value.
15    */
16    void FilteringSignal(Filter* filt);
```

```
1   ComplexSignal* ComplexSignal::Filtering(Filter* filt)
2   {
3       /** Declaring the variables of the filter and getting their values */
4       int Nfir;
5       int Niir;
6       double* b = NULL;
7       double* a = NULL;
8       double* tabsreal = NULL;
9       double* tabsimag = NULL;
10
11      Nfir = filt->GetNfir();
12      Niir = filt->GetNiir();
13      b = filt->Getb();
14      a = filt->Geta();
15      tabsreal = filt->GetTabsReal();
16      tabsimag = filt->GetTabsImag();
17
18      int maxN = Niir;
19
20      if(Nfir>Niir)
21          maxN = Nfir;
22
23      //Allocate memory
24      double* vreal = NULL;
25      vreal = (double*) calloc(maxN+1, sizeof(double));
26      if (NULL==vreal)
27      {
28          std::cerr << "ERROR: Could not allocate memory for vreal." <<
                std::endl;
29          exit(EXIT_FAILURE);
```

```
30      }
31
32      double* vimag = NULL;
33      vimag = (double*) calloc(maxN+1, sizeof(double));
34      if (NULL==vimag)
35      {
36          std::cerr << "ERROR: Could not allocate memory for vimag." <<
                std::endl;
37          exit(EXIT_FAILURE);
38      }
39
40      double* real_filtered = NULL;
41      real_filtered = (double*) calloc(m_length, sizeof(double));
42      if (NULL==real_filtered)
43      {
44          std::cerr << "ERROR: Could not allocate memory for real_filtered."
                << std::endl;
45          exit(EXIT_FAILURE);
46      }
47
48      double* imag_filtered = NULL;
49      imag_filtered = (double*) calloc(m_length, sizeof(double));
50      if (NULL==imag_filtered)
51      {
52          std::cerr << "ERROR: Could not allocate memory for imag_filtered."
                << std::endl;
53          exit(EXIT_FAILURE);
54      }
55
56      /*Filtering of the real and imaginary parts*/
57      // Init v[n]
58      vreal[0] = m_real[0]/a[0];
59      vimag[0] = m_imag[0]/a[0];
60
61      for(int k=1; k<=maxN; k++)
62      {
63          vreal[k] = tabsreal[k-1];
64          vimag[k] = tabsimag[k-1];
65      }
66
67
68      //std::cout << maxN << std::endl;
69      for(int n=0; n<m_length; n++)
70      {
71          real_filtered[n] = b[0]*vreal[0];
72          imag_filtered[n] = b[0]*vimag[0];
73
74          for(int k=1; k<=maxN; k++)
75          {
76              real_filtered[n] = real_filtered[n] + b[k]*vreal[k];
77              imag_filtered[n] = imag_filtered[n] + b[k]*vimag[k];
```

```
78          }
79
80          // Update the taps content
81          for(int k=maxN; k>0; k--)
82          {
83              vreal[k] = vreal[k-1];
84              vimag[k] = vimag[k-1];
85          }
86          ////////////////////
87
88          vreal[0] = 0;
89          vimag[0] = 0;
90
91          for(int k=1; k<=maxN; k++)
92          {
93              vreal[0] = vreal[0] + vreal[k]*(-a[k]);
94              vimag[0] = vimag[0] + vimag[k]*(-a[k]);
95          }
96
97          vreal[0] = (vreal[0] + m_real[n+1])*(1/a[0]);
98          vimag[0] = (vimag[0] + m_imag[n+1])*(1/a[0]);
99
100     }
101
102     printf("Creating NEW signal filtered\n");
103
104     ComplexSignal* signal_filtered = new ComplexSignal(m_length, (const
            double*) real_filtered, (const double*) imag_filtered, m_Fs,
            m_initial_sample);
105
106     // Set the new tabs of the filter
107
108     filt->SetFIR(Nfir, b);
109     filt->SetIIR(Niir, a);
110     filt->SetTabs(maxN+1, vreal, vimag);
111     /*
112     printf("Final taps:\n");
113     for(int h=0; h<=maxN; h++)
114     {
115         printf(" (%3.2f)i + (%3.2f)j",vreal[h], vimag[h]);
116     }
117     printf("\n");
118     */
119
120     free(vreal);
121     free(vimag);
122     free(real_filtered);
123     free(imag_filtered);
124
125     return signal_filtered;
126
```

```
127 }
128
129 void ComplexSignal::FilteringSignal(Filter* filt)
130 {
131     /** Declaring the variables of the filter and getting their values */
132     int Nfir;
133     int Niir;
134     double* b = NULL;
135     double* a = NULL;
136     double* tabsreal = NULL;
137     double* tabsimag = NULL;
138
139     Nfir = filt->GetNfir();
140     Niir = filt->GetNiir();
141     b = filt->Getb();
142     a = filt->Geta();
143     tabsreal = filt->GetTabsReal();
144     tabsimag = filt->GetTabsImag();
145
146     int maxN = Niir;
147
148     if(Nfir>Niir)
149         maxN = Nfir;
150
151     //Allocate memory
152     double* vreal = NULL;
153     vreal = (double*) calloc(maxN+1, sizeof(double));
154     if (NULL==vreal)
155     {
156         std::cerr << "ERROR: Could not allocate memory for vreal." <<
                std::endl;
157         exit(EXIT_FAILURE);
158     }
159
160     double* vimag = NULL;
161     vimag = (double*) calloc(maxN+1, sizeof(double));
162     if (NULL==vimag)
163     {
164         std::cerr << "ERROR: Could not allocate memory for vimag." <<
                std::endl;
165         exit(EXIT_FAILURE);
166     }
167
168     double* real_filtered = NULL;
169     real_filtered = (double*) calloc(m_length, sizeof(double));
170     if (NULL==real_filtered)
171     {
172         std::cerr << "ERROR: Could not allocate memory for real_filtered."
                << std::endl;
173         exit(EXIT_FAILURE);
174     }
```

```
175
176     double* imag_filtered = NULL;
177     imag_filtered = (double*) calloc(m_length, sizeof(double));
178     if (NULL==imag_filtered)
179     {
180         std::cerr << "ERROR: Could not allocate memory for imag_filtered."
                << std::endl;
181         exit(EXIT_FAILURE);
182     }
183
184     /*Filtering of the real and imaginary parts*/
185     // Init v[n]
186     vreal[0] = m_real[0]/a[0];
187     vimag[0] = m_imag[0]/a[0];
188
189     for(int k=1; k<=maxN; k++)
190     {
191         vreal[k] = tabsreal[k-1];
192         vimag[k] = tabsimag[k-1];
193     }
194
195
196     //std::cout << maxN << std::endl;
197     for(int n=0; n<m_length; n++)
198     {
199         real_filtered[n] = b[0]*vreal[0];
200         imag_filtered[n] = b[0]*vimag[0];
201
202         for(int k=1; k<=maxN; k++)
203         {
204             real_filtered[n] = real_filtered[n] + b[k]*vreal[k];
205             imag_filtered[n] = imag_filtered[n] + b[k]*vimag[k];
206         }
207
208         // Update the taps content
209         for(int k=maxN; k>0; k--)
210         {
211             vreal[k] = vreal[k-1];
212             vimag[k] = vimag[k-1];
213         }
214         ////////////////////
215
216         vreal[0] = 0;
217         vimag[0] = 0;
218
219         for(int k=1; k<=maxN; k++)
220         {
221             vreal[0] = vreal[0] + vreal[k]*(-a[k]);
222             vimag[0] = vimag[0] + vimag[k]*(-a[k]);
223         }
224
```

```
225         vreal[0] = (vreal[0] + m_real[n+1])*(1/a[0]);
226         vimag[0] = (vimag[0] + m_imag[n+1])*(1/a[0]);
227 
228     }
229 
230 //    std::cout << "Filtered signal" << std::endl;
231 
232     //ComplexSignal* signal_filtered = new ComplexSignal(m_length, (const
            double*) real_filtered, (const double*) imag_filtered, m_Fs,
            m_initial_sample);
233 
234     //Set the new values of the complex signal.
235     SetArray(m_length, (const double*) real_filtered, (const double*)
            imag_filtered);
236 
237     // Set the new tabs of the filter
238 
239     filt->SetFIR(Nfir, b);
240     filt->SetIIR(Niir, a);
241     filt->SetTabs(maxN+1, vreal, vimag);
242     /*
243     printf("Final taps:\n");
244     for(int h=0; h<=maxN; h++)
245     {
246         printf(" (%3.2f)i + (%3.2f)j",vreal[h], vimag[h]);
247     }
248     printf("\n");
249     */
250 
251     free(vreal);
252     free(vimag);
253 
254 
255     free(real_filtered);
256     free(imag_filtered);
257 
258 }
```

D.1.2. Main programs

D.1.2.1. Compute Spectrum

```
1
2  #include <iostream>
3
4  #include <complex_signal.h>
5  #include <gsl/gsl_complex_math.h>
6  #include <gsl/gsl_fft_complex.h>
7  #include <mgl2/mgl.h>
8  #include <filter.h>
9  #include <spectrum.h>
10
11
12 using namespace std;
13
14 int main(int argc, char* argv[])
15 {
16
17   cout << "Example of Spectrum program with a BIBASPIR FILE (L1)" << endl;
18
19   ComplexSignal* supnocrop;
20   ComplexSignal* sdwnocrop;
21
22   // Read signal from BIBASPIR file
23   string bibafilename="17nov2017";
24   supnocrop = ComplexSignal::GetFromBIBASPIRFile(bibafilename, 1, 1);
25   sdwnocrop = ComplexSignal::GetFromBIBASPIRFile(bibafilename, 0, 1);
26   /*
27   ComplexWaveform* Pnocrop;
28
29   printf("Computing CROSS-correlation\n");
30   Pnocrop = ComplexSignal::CorrelationInTimeDomain(sdwnocrop, supnocrop,
         80000, 80000, -10, 1, 150);
31   Pnocrop->PlotMag("L1Cross-Correlation");
32
33   delete(Pnocrop);
34   */
35   //Crop the signal
36   printf("Crop signal\n");
37   ComplexSignal* sup1;
38   ComplexSignal* sdw1;
39
40   sup1 = ComplexSignal::Crop(supnocrop, 0, 8e5);
41   sdw1 = ComplexSignal::Crop(sdwnocrop, 0, 8e5);
42
43   delete(supnocrop);
44   delete(sdwnocrop);
45
46   // DC Calibration
```

```
47  gsl_complex mean_up;
48  gsl_complex mean_dw;
49
50  mean_dw = sdw1->Mean();
51  mean_up = sup1->Mean();
52
53  sup1->Subtract(mean_up);// NO crea una nueva ComplexSignal
54  sdw1->Subtract(mean_dw);
55
56  // Compute spectrum of signals without multiplying the phasor
57  printf("Compute spectrum of signals without multiplying the phasor\n");
58  Spectrum* spect_sup1;
59  Spectrum* spect_sdw1;
60
61  spect_sup1 = sup1->Fourier();
62  spect_sdw1 = sdw1->Fourier();
63
64  // Plot spectrum of signals
65  printf("Plot spectrum of signals\n");
66  spect_sup1->PlotMag("L1supSpectrumCalibrated");
67  spect_sdw1->PlotMag("L1sdwSpectrumCalibrated");
68
69  // Creating the phasor
70  printf("Creating the phasor\n");
71  ComplexSignal* phasor;
72
73  phasor = ComplexSignal::Phasor(7e6, 80e6, sup1->GetLength(), 0);
74
75  // Multiplying signals
76  printf("Multiplying signals\n");
77  ComplexSignal* sup;
78  ComplexSignal* sdw;
79
80  sup = ComplexSignal::Multiply(sup1, phasor);
81  sdw = ComplexSignal::Multiply(sdw1, phasor);
82
83  delete(phasor);
84
85  // Compute spectrum of signals
86  printf("Compute spectrum of signals after multiplying phasors\n");
87  Spectrum* spect_sup;
88  Spectrum* spect_sdw;
89
90  spect_sup = sup->Fourier();
91  spect_sdw = sdw->Fourier();
92
93  // Plot spectrum of signals
94  printf("Plot spectrum of signals\n");
95  spect_sup->PlotMag("L1supSpectrumCalibratedPhasor");
96  spect_sdw->PlotMag("L1sdwSpectrumCalibratedPhasor");
97
```

```
98  // Filtering
99
100 Filter* filtup;
101 Filter* filtdw;
102
103 string filename = "Butterworth10MHz";
104
105 filtup = Filter::GetFromFilterFile(filename);
106 filtdw = Filter::GetFromFilterFile(filename);
107
108 ComplexSignal* sup_filtered;
109 ComplexSignal* sdw_filtered;
110
111
112 std::cout << "Initial values of UP and DW filters" << std::endl;
113 filtup->Print();
114 filtdw->Print();
115
116 sup_filtered = sup->Filtering(filtup);
117 sdw_filtered = sdw->Filtering(filtdw);
118
119
120 std::cout << "Final values of UP and DW filterS" << std::endl;
121 filtup->Print();
122 filtdw->Print();
123
124 delete(filtup);
125 delete(filtdw);
126
127 //sup_filtered->PlotReIm("ZZFilterParamBiba10",0, 2000);
128
129 // Compute spectrum of filtered signals
130 printf("Compute spectrum of filtered signals\n");
131 Spectrum* spect_supfiltered;
132 Spectrum* spect_sdwfiltered;
133
134 spect_supfiltered = sup_filtered->Fourier();
135 spect_sdwfiltered = sdw_filtered->Fourier();
136
137 // Plot spectrum of filtered signals
138 printf("Plot spectrum of filtered signals\n");
139 spect_supfiltered->PlotMag("L1sup_filteredSpectrumCalibratedPhasor");
140 spect_sdwfiltered->PlotMag("L1sdw_filteredSpectrumCalibratedPhasor");
141
142 // Cross-Correlation between UP and DW
143 /*
144 ComplexWaveform* P;
145
146 printf("Computing CROSS-correlation with filtering in UP and DW
      signals\n");
```

```
147   P = ComplexSignal::CorrelationInTimeDomain(sdw_filtered, sup_filtered,
          80000, 80000, -10, 1, 150);
148   P->PlotMag("L1Cross-Correlation_filteredCalibratedPhasor");
149
150   delete(P);
151   */
152   free(sup);
153   free(sdw);
154   free(sup_filtered);
155   free(sdw_filtered);
156   free(spect_sup);
157   free(spect_sdw);
158   free(spect_supfiltered);
159   free(spect_sdwfiltered);
160   free(sup1);
161   free(sdw1);
162
163   return 0;
164 }
```

D.1.2.2. Waveform Cluster

```
// Read a BIBA-SPIR file and generate waveform_cluster
//BALLOON

#include <iostream>
#include <sstream>

#include <complex_signal.h>
#include <filter.h>
#include <spectrum.h>

int main(void)
{
  ComplexSignal* x;
  ComplexSignal* y;
  //std::string filename="../../../../rain/30Sec";
  std::string filename="2-16deg-19Sec";

  // Reading signals from File
  std::cout << "Reading BIBA-SPIR file (L1)." << std::endl;

  x = ComplexSignal::GetFromBIBASPIRFile(filename, 1, 1);
  y = ComplexSignal::GetFromBIBASPIRFile(filename, 0, 1);

  // DC Calibration
  std::cout << "DC Calibration." << std::endl;
  gsl_complex xmean = x->Mean();
  gsl_complex ymean = y->Mean();
  std::cout << "mean x: " << GSL_REAL(xmean) << "j" << GSL_IMAG(xmean) <<
      std::endl;
  x->Subtract(xmean);
  std::cout << "mean y: " << GSL_REAL(ymean) << "j" << GSL_IMAG(ymean) <<
      std::endl;
  y->Subtract(ymean);

  // Creating the phasor
  std::cout << "Creating the phasor." << std::endl;
  ComplexSignal* phasor = ComplexSignal::Phasor(7e6, 80e6, y->GetLength(),
      0);

  // Multiplying signals
  std::cout << "Shift signals to baseband." << std::endl;
  //ComplexSignal* xlo = ComplexSignal::Multiply(x, phasor);
  x->Multiply(phasor);

  //delete(x);

  //ComplexSignal* ylo = ComplexSignal::Multiply(y, phasor);
  y->Multiply(phasor);

```

```
47  //delete(y);
48  delete(phasor);
49
50  // Filtering
51  //std::string filter_file = "10butterworth12e6";
52  std::string filter_file = "Butterworth10MHz";
53  std::cout << "Filtering with " << filter_file << std::endl;
54
55  Filter* filtx = Filter::GetFromFilterFile(filter_file);
56  std::cout << "a" << std::endl;
57  Filter* filty = Filter::GetFromFilterFile(filter_file);
58  std::cout << "b" << std::endl;
59  /*
60  std::cout << "Initial values of UP and DW filters:" << std::endl;
61  filtx->Print();
62  filty->Print();
63  */
64  //ComplexSignal* x_filtered = xlo->Filtering(filtx);
65  //ComplexSignal* y_filtered = ylo->Filtering(filty);
66
67  //ComplexSignal* x_filtered = x->Filtering(filtx);
68  //ComplexSignal* y_filtered = y->Filtering(filty);
69  std::cout << "Ha creado el filtro." << std::endl;
70
71  x->FilteringSignal(filtx);
72  y->FilteringSignal(filty);
73
74  //delete(xlo);
75  //delete(ylo);
76
77  //delete(x);
78  //delete(y);
79
80  /*
81  std::cout << "Final values of UP and DW filterS" << std::endl;
82  filtx->Print();
83  filty->Print();
84  */
85  delete(filtx);
86  delete(filty);
87
88  // Compute correlation clusters
89  int L = 999;
90  //int W = 301;
91
92  unsigned int N=80000;
93  unsigned int n0=3015;
94  int          min_m=0;
95  unsigned int step_m=1;
96  int          max_m=20;
97
```

```
98    std::cout << "Computing cluster..." << std::endl;
99    Waveform_complex_cluster* R;
100
101   R = ComplexSignal::CorrelationInTimeDomain(y,x,N,n0, min_m, step_m,
          max_m, L);
102   R->print(0);
103   std::cout << "Plotting cluster." << std::endl;
104   R->PlotMag(filename + "_R-mag_N=" + std::to_string(N) + " n0=" +
          std::to_string(n0), min_m, max_m);
105   R->PlotPower(filename + "_R-power_N=" + std::to_string(N) + " n0=" +
          std::to_string(n0), min_m, max_m);
106   R->PlotPhase(filename + "_R-phase-deg_N=" + std::to_string(N) + " n0=" +
          std::to_string(n0), min_m, max_m, 'd');
107
108   delete(R);
109
110   return 0;
111 }
```

D.1.2.3. Delay Doppler Map

```
// Read a BIBA-SPIR file and calculates the doppler shift
//BALLOON

#include <iostream>
#include <sstream>
#include <iomanip>

#include <complex_signal.h>
#include <filter.h>
#include <spectrum.h>
#include <ddm.h>

int main(void)
{
  ComplexSignal* x;
  ComplexSignal* y;
  ComplexSignal* lo;


  std::string filename="2-16deg-18Sec";

        // READING SIGNALS FROM BIBASPIRFILE
        // L Channels: 1-> L1 and 0->L5
        unsigned int L1_nL5 = 1;
        if(L1_nL5 == 1)
        {
            std::cout << "Reading BIBA-SPIR file (L1)" << std::endl;
        }
        else
        {
            std::cout << "Reading BIBA-SPIR file (L5)" << std::endl;
        }

        x = ComplexSignal::GetFromBIBASPIRFile(filename, 1, L1_nL5);
        y = ComplexSignal::GetFromBIBASPIRFile(filename, 0, L1_nL5);

        // DC CALIBRATION
        std::cout << "Calibrating" << std::endl;
        gsl_complex xmean = x->Mean();
        gsl_complex ymean = y->Mean();
        std::cout << "mean x: " << GSL_REAL(xmean) << "j" <<
            GSL_IMAG(xmean) << std::endl;
        x->Subtract(xmean);
        std::cout << "mean y: " << GSL_REAL(ymean) << "j" <<
            GSL_IMAG(ymean) << std::endl;
        y->Subtract(ymean);

        // SHIFT SIGNALS TO BASEBAND
        std::cout << "Shifting signals to baseband" << std::endl;
```

```
48          double f_lo; // frequency of the phasor
49          if(L1_nL5 == 1)
50          {
51              // L1 Channel
52              f_lo = 7e6;
53          }
54          else
55          {
56              // L5 Channel
57              f_lo = -9.75e6;
58          }
59          std::cout << "Getting Phasor with f = " << f_lo << std::endl;
60          lo = ComplexSignal::Phasor(f_lo, x->GetFs(), x->GetLength(), 0);
61          std::cout << "Shifting to baseband" << std::endl;
62          x->Multiply(lo);
63          y->Multiply(lo);
64          delete(lo);
65
66          // FILTERING
67          std::string filter_file = "Butterworth10MHz";
68          std::cout << "Filtering with " << filter_file << std::endl;
69
70          Filter* filtx = Filter::GetFromFilterFile(filter_file);
71          Filter* filty = Filter::GetFromFilterFile(filter_file);
72
73          x->FilteringSignal(filtx);
74          y->FilteringSignal(filty);
75
76          delete(filtx);
77          delete(filty);
78
79
80          //std::cout << "Computing waveform cluster." << std::endl;
81          //unsigned int L=999;
82          unsigned int N=8e6; // 1 plot DDM para cada 100 ms
83          unsigned int n0=15+9*N; // aqui al principio sera n0=x+0; luego
                n0=x+N; n0=x+2; n0=x+3N; n0=x+4N; ... hasta 10N
84          int         min_m=0;
85          unsigned int step_m=1;
86          int         max_m=+15;
87
88          std::cout << "Computing DDM" << std::endl;
89          double min_f = -40;
90          double step_f = 1;
91          double max_f = +40;
92          DDM* ddm;
93          ddm = ComplexSignal::DDMInTimeDomain(y, x, N, n0, min_m, step_m,
                max_m, min_f, step_f, max_f);
94          std::string plotmag_name = filename +"_DDM-mag_N=" +
                std::to_string(N) + " n0=" + std::to_string(n0);
95          ddm->PlotMag(plotmag_name, min_m, max_m);
```

```
96          delete(x);
97          delete(y);
98          delete(ddm);
99
100         return 0;
101 }
```

D.2. Python GUI program

```
import wx
import wx.xrc
import sys
import os
import os.path

wildcard = ""

##### DEFAULT VALUES #####
files_value = "No selected files"
L1button_value = True;
L5button_value = False;

cal_on_value = True
cal_off_value = False

hx_on_value = True
hx_off_value = False
hy_on_value = True
hy_off_value = False
filter_file_value = ""

fxmin_value = "0"
fxstep_value = "1"
fxmax_value = "1000"

# 0 --> UP, 1 --> DW
x_choice_value = 0
y_choice_value = 1

N_value = "80000"
n0_value = "80000"
w_value = "998"
mmin_value = "-10"
mstep_value = "1"
mmax_value = "150"

opt_plot_value = True
opt_csv_value = False
opt_waveform_value = True
opt_doppler_value = False

#########################

#### MainFrame class
    ###########################################################################
##############################################################################################
class MainFrame ( wx.Frame ):
```

```
48
49      def __init__(self):
50          wx.Frame.__init__ ( self, None, title = u"Signals of Opportunity
                Processing GUI", pos = wx.DefaultPosition, size = wx.Size(
                640,540 ), style = wx.DEFAULT_FRAME_STYLE|wx.TAB_TRAVERSAL )
51
52          ##### Color and default size of the MainFrame
                ########################################
53          self.SetSizeHintsSz( wx.DefaultSize, wx.DefaultSize )
54          self.SetBackgroundColour( wx.Colour( 194, 219, 245 ) )
55          ##############################################################################
56
57          ##### CREATE THE MENU
58          self.MainBar = wx.MenuBar( 0 )
59          self.MenuFile = wx.Menu()
60          self.MenuClose = wx.MenuItem( self.MenuFile, wx.ID_EXIT,
                "&Exit\tAlt-q", wx.EmptyString, wx.ITEM_NORMAL )
61          self.Bind(wx.EVT_MENU, self.OnTimeToClose, id=wx.ID_EXIT)
62          self.MenuFile.AppendItem( self.MenuClose )
63
64          self.MainBar.Append( self.MenuFile, "&File" )
65
66          self.MenuOptions = wx.Menu()
67          self.MainBar.Append( self.MenuOptions, "&Options" )
68
69          self.MenuHelp = wx.Menu()
70          self.MainBar.Append( self.MenuHelp, "&Help" )
71
72          self.SetMenuBar( self.MainBar )
73          self.Centre( wx.BOTH )
74
75          ##############################################################################
76          ##############################################################################
77          ##############################################################################
78
79          ##### SIZER OF THE MAIN FRAME
                ########################################################
80          MainSizer = wx.BoxSizer( wx.HORIZONTAL )
81          ##############################################################################
82
83
84          ##### CREATE INPUTS NOTEBOOK
                ########################################################
85          self.InputsNb = wx.Notebook( self, wx.ID_ANY, wx.DefaultPosition,
                wx.DefaultSize, 0 )
86          ##############################################################################
87
88          ##### CREATE FILE TAB
                ################################################################
89          self.FileTab = wx.ScrolledWindow( self.InputsNb, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.HSCROLL|wx.VSCROLL )
```

```
90          self.FileTab.SetScrollRate( 5, 5 )
91          self.FileTab.SetBackgroundColour( wx.Colour( 194, 219, 245 ) )
92
93          FileSizer = wx.BoxSizer( wx.VERTICAL )
94
95          self.m_staticText2 = wx.StaticText( self.FileTab, wx.ID_ANY,
                u"Import Files", wx.DefaultPosition, wx.DefaultSize, 0 )
96          self.m_staticText2.Wrap( -1 )
97          self.m_staticText2.SetFont( wx.Font( 10, 74, 90, 92, False, "Arial"
                ) )
98
99          FileSizer.Add( self.m_staticText2, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
100
101         self.browse = wx.Button( self.FileTab, wx.ID_ANY, u"Browse",
                wx.DefaultPosition, wx.DefaultSize, 0 )
102         FileSizer.Add( self.browse, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
103
104         self.m_staticText3 = wx.StaticText( self.FileTab, wx.ID_ANY, u"(More
                than 1 file can be chosen, Ctrl. + Select)", wx.DefaultPosition,
                wx.DefaultSize, 0 )
105         self.m_staticText3.Wrap( -1 )
106         self.m_staticText3.SetFont( wx.Font( 10, 74, 93, 90, False, "Arial
                Narrow" ) )
107
108         FileSizer.Add( self.m_staticText3, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
109
110         self.m_staticText4 = wx.StaticText( self.FileTab, wx.ID_ANY,
                u"Selected files:", wx.DefaultPosition, wx.DefaultSize, 0 )
111         self.m_staticText4.Wrap( -1 )
112         FileSizer.Add( self.m_staticText4, 0, wx.ALL, 5 )
113
114         self.files = wx.TextCtrl( self.FileTab, wx.ID_ANY, files_value,
                style=wx.TE_MULTILINE, size = (100,200))
115
116         FileSizer.Add( self.files, 0, wx.ALIGN_CENTER|wx.ALL|wx.EXPAND, 5 )
117
118         self.Lband = wx.StaticText( self.FileTab, wx.ID_ANY, u"Select a
                frequency band:", wx.DefaultPosition, wx.DefaultSize, 0 )
119         self.Lband.Wrap( -1 )
120         FileSizer.Add( self.Lband, 0, wx.ALL, 5 )
121
122         self.PanelLband = wx.Panel( self.FileTab, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.TAB_TRAVERSAL )
123         SizerLband = wx.BoxSizer( wx.HORIZONTAL )
124
125         self.L1button = wx.RadioButton( self.PanelLband, wx.ID_ANY, u"L1",
                wx.DefaultPosition, wx.DefaultSize, 0 )
126         SizerLband.Add( self.L1button, 0, wx.ALL, 5 )
127
128         self.L5button = wx.RadioButton( self.PanelLband, wx.ID_ANY, u"L5",
                wx.DefaultPosition, wx.DefaultSize, 0 )
```

```
129         SizerLband.Add( self.L5button, 0, wx.ALL, 5 )
130
131
132         self.PanelLband.SetSizer( SizerLband )
133         self.PanelLband.Layout()
134         SizerLband.Fit( self.PanelLband )
135         FileSizer.Add( self.PanelLband, 1, wx.EXPAND |wx.ALL, 5 )
136
137         self.FileTab.SetSizer( FileSizer )
138         self.FileTab.Layout()
139         FileSizer.Fit( self.FileTab )
140         ##########################################################################################
141         ##### CREATE CALIBRATION, FILTERING AND DOPPLER TAB
                ##################################
142         self.CFDTab = wx.ScrolledWindow( self.InputsNb, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.HSCROLL|wx.VSCROLL )
143         self.CFDTab.SetScrollRate( 5, 5 )
144         self.CFDTab.SetBackgroundColour( wx.Colour( 194, 219, 245 ) )
145
146         CFDSizer = wx.BoxSizer( wx.VERTICAL )
147
148         self.m_staticText5 = wx.StaticText( self.CFDTab, wx.ID_ANY, u"DC
                Calibration:", wx.DefaultPosition, wx.DefaultSize, 0 )
149         self.m_staticText5.Wrap( -1 )
150         self.m_staticText5.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial" )
                )
151
152         CFDSizer.Add( self.m_staticText5, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
153
154         self.CalibrationPanel = wx.Panel( self.CFDTab, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.TAB_TRAVERSAL )
155         bSizer5 = wx.BoxSizer( wx.HORIZONTAL )
156
157         self.cal_on = wx.RadioButton( self.CalibrationPanel, wx.ID_ANY,
                u"ON", wx.DefaultPosition, wx.DefaultSize, 0 )
158         self.cal_on.SetValue( cal_on_value )
159         bSizer5.Add( self.cal_on, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
160
161         self.cal_off = wx.RadioButton( self.CalibrationPanel, wx.ID_ANY,
                u"OFF", wx.DefaultPosition, wx.DefaultSize, 0 )
162         self.cal_off.SetValue( not cal_on_value )
163         bSizer5.Add( self.cal_off, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
164
165
166         self.CalibrationPanel.SetSizer( bSizer5 )
167         self.CalibrationPanel.Layout()
168         bSizer5.Fit( self.CalibrationPanel )
169         CFDSizer.Add( self.CalibrationPanel, 1, wx.ALIGN_LEFT|wx.ALL, 5 )
170
171         self.m_staticText6 = wx.StaticText( self.CFDTab, wx.ID_ANY,
                u"Filtering:", wx.DefaultPosition, wx.DefaultSize, 0 )
```

```
172         self.m_staticText6.Wrap( -1 )
173         self.m_staticText6.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial" )
                )
174
175         CFDSizer.Add( self.m_staticText6, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
176
177         self.m_panel10 = wx.Panel( self.CFDTab, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.TAB_TRAVERSAL )
178         bSizer22 = wx.BoxSizer( wx.HORIZONTAL )
179
180         self.m_panel12 = wx.Panel( self.m_panel10, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.TAB_TRAVERSAL )
181         bSizer23 = wx.BoxSizer( wx.VERTICAL )
182
183         self.HxPanel = wx.Panel( self.m_panel12, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.TAB_TRAVERSAL )
184         bSizer6 = wx.BoxSizer( wx.HORIZONTAL )
185
186         self.m_staticText7 = wx.StaticText( self.HxPanel, wx.ID_ANY,
                u"hx[n]", wx.DefaultPosition, wx.DefaultSize, 0 )
187         self.m_staticText7.Wrap( -1 )
188         bSizer6.Add( self.m_staticText7, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
189
190         self.hx_on = wx.RadioButton( self.HxPanel, wx.ID_ANY, u"ON",
                wx.DefaultPosition, wx.DefaultSize, 0 )
191         self.hx_on.SetValue( True )
192         bSizer6.Add( self.hx_on, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
193
194         self.hx_off = wx.RadioButton( self.HxPanel, wx.ID_ANY, u"OFF",
                wx.DefaultPosition, wx.DefaultSize, 0 )
195         bSizer6.Add( self.hx_off, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
196
197
198         self.HxPanel.SetSizer( bSizer6 )
199         self.HxPanel.Layout()
200         bSizer6.Fit( self.HxPanel )
201         bSizer23.Add( self.HxPanel, 1, wx.ALIGN_LEFT|wx.ALL, 5 )
202
203         ###
204
205
206
207         ###
208
209         self.HyPanel = wx.Panel( self.m_panel12, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.TAB_TRAVERSAL )
210         bSizer7 = wx.BoxSizer( wx.HORIZONTAL )
211
212         self.m_staticText8 = wx.StaticText( self.HyPanel, wx.ID_ANY,
                u"hy[n]", wx.DefaultPosition, wx.DefaultSize, 0 )
213         self.m_staticText8.Wrap( -1 )
```

```
214         bSizer7.Add( self.m_staticText8, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
215
216         self.hy_on = wx.RadioButton( self.HyPanel, wx.ID_ANY, u"ON",
                wx.DefaultPosition, wx.DefaultSize, 0 )
217         self.hy_on.SetValue( True )
218         bSizer7.Add( self.hy_on, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
219
220         self.hy_off = wx.RadioButton( self.HyPanel, wx.ID_ANY, u"OFF",
                wx.DefaultPosition, wx.DefaultSize, 0 )
221         bSizer7.Add( self.hy_off, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
222
223
224         self.HyPanel.SetSizer( bSizer7 )
225         self.HyPanel.Layout()
226         bSizer7.Fit( self.HyPanel )
227         bSizer23.Add( self.HyPanel, 1, wx.ALIGN_LEFT|wx.ALL, 5 )
228
229         self.m_panel12.SetSizer( bSizer23 )
230         self.m_panel12.Layout()
231         bSizer23.Fit( self.m_panel12 )
232         bSizer22.Add( self.m_panel12, 1, wx.EXPAND |wx.ALL, 5 )
233
234         bSizer24 = wx.BoxSizer (wx.VERTICAL)
235         self.filterbtn = wx.Button( self.m_panel10, wx.ID_ANY, u"Select
                filter parameters", wx.DefaultPosition, wx.DefaultSize, 0 )
236         bSizer24.Add( self.filterbtn, 0, wx.ALL, 5 )
237
238         self.filter_file = wx.TextCtrl( self.m_panel10, wx.ID_ANY,
                filter_file_value, wx.DefaultPosition, wx.Size(150, -1), 0 )
239         bSizer24.Add( self.filter_file, 0, wx.ALL, 5 )
240
241
242         bSizer22.Add( bSizer24, 1, wx.ALIGN_CENTER, 5 )
243
244
245         self.m_panel10.SetSizer( bSizer22 )
246         self.m_panel10.Layout()
247         bSizer22.Fit( self.m_panel10 )
248         CFDSizer.Add( self.m_panel10, 2, wx.EXPAND |wx.ALL, 5 )
249
250         self.m_staticText9 = wx.StaticText( self.CFDTab, wx.ID_ANY,
                u"Scanning frequencies for Doppler computation:",
                wx.DefaultPosition, wx.DefaultSize, 0 )
251         self.m_staticText9.Wrap( -1 )
252         self.m_staticText9.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial" )
                )
253
254         CFDSizer.Add( self.m_staticText9, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
255
256         FreqSizer1 = wx.BoxSizer( wx.HORIZONTAL )
257
```

```
258         self.m_staticText10 = wx.StaticText( self.CFDTab, wx.ID_ANY,
                u"fxmin[Hz] = ", wx.DefaultPosition, wx.DefaultSize, 0 )
259         self.m_staticText10.Wrap( -1 )
260         FreqSizer1.Add( self.m_staticText10, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
261
262         self.fxmin = wx.TextCtrl( self.CFDTab, wx.ID_ANY, fxmin_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
263         FreqSizer1.Add( self.fxmin, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
264
265
266         CFDSizer.Add( FreqSizer1, 1, wx.ALIGN_LEFT, 5 )
267
268         FreqSizer2 = wx.BoxSizer( wx.HORIZONTAL )
269
270         self.m_staticText11 = wx.StaticText( self.CFDTab, wx.ID_ANY,
                u"fxstep[Hz] = ", wx.DefaultPosition, wx.DefaultSize, 0 )
271         self.m_staticText11.Wrap( -1 )
272         FreqSizer2.Add( self.m_staticText11, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
273
274         self.fxstep = wx.TextCtrl( self.CFDTab, wx.ID_ANY, fxstep_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
275         FreqSizer2.Add( self.fxstep, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
276
277
278         CFDSizer.Add( FreqSizer2, 1, wx.ALIGN_LEFT, 5 )
279
280         FreqSizer3 = wx.BoxSizer( wx.HORIZONTAL )
281
282         self.m_staticText12 = wx.StaticText( self.CFDTab, wx.ID_ANY,
                u"fxmax[Hz] = ", wx.DefaultPosition, wx.DefaultSize, 0 )
283         self.m_staticText12.Wrap( -1 )
284         FreqSizer3.Add( self.m_staticText12, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
285
286         self.fxmax = wx.TextCtrl( self.CFDTab, wx.ID_ANY, fxmax_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
287         FreqSizer3.Add( self.fxmax, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
288
289
290         CFDSizer.Add( FreqSizer3, 1, wx.ALIGN_LEFT, 5 )
291
292         self.CFDTab.SetSizer( CFDSizer )
293         self.CFDTab.Layout()
294         CFDSizer.Fit( self.CFDTab )
295         ##########################################################################################
296
297         ##### CREATE CORRELATION TAB
                ########################################################
298         self.CorrelationTab = wx.ScrolledWindow( self.InputsNb, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.HSCROLL|wx.VSCROLL )
299         self.CorrelationTab.SetScrollRate( 5, 5 )
300         self.CorrelationTab.SetBackgroundColour( wx.Colour( 194, 219, 245 ) )
```

```
301
302         CorrelationSizer = wx.BoxSizer( wx.VERTICAL )
303
304         self.m_staticText13 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"Correlation functions:", wx.DefaultPosition, wx.DefaultSize, 0
                )
305         self.m_staticText13.Wrap( -1 )
306         self.m_staticText13.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial"
                ) )
307
308         CorrelationSizer.Add( self.m_staticText13, 0, wx.ALIGN_LEFT|wx.ALL,
                5 )
309
310         XYSizer = wx.BoxSizer( wx.HORIZONTAL )
311
312         self.m_staticText181 = wx.StaticText( self.CorrelationTab,
                wx.ID_ANY, u"X =", wx.DefaultPosition, wx.DefaultSize, 0 )
313         self.m_staticText181.Wrap( -1 )
314         XYSizer.Add( self.m_staticText181, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
315
316         x_choiceChoices = ['UP','DW']
317         self.x_choice = wx.Choice( self.CorrelationTab, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, x_choiceChoices, 0)
318         self.x_choice.SetSelection( x_choice_value )
319         XYSizer.Add( self.x_choice, 0, wx.ALL, 5 )
320
321         self.m_staticText18 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"Y =", wx.DefaultPosition, wx.DefaultSize, 0 )
322         self.m_staticText18.Wrap( -1 )
323         XYSizer.Add( self.m_staticText18, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
324
325         y_choiceChoices = x_choiceChoices
326         self.y_choice = wx.Choice( self.CorrelationTab, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, y_choiceChoices, 1 )
327         self.y_choice.SetSelection( y_choice_value )
328         XYSizer.Add( self.y_choice, 0, wx.ALL, 5 )
329
330
331         CorrelationSizer.Add( XYSizer, 1, wx.ALIGN_LEFT|wx.ALL, 5 )
332
333         self.cross_auto = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"CROSS - Correlation will be performed", wx.DefaultPosition,
                wx.DefaultSize, 0 )
334         self.cross_auto.Wrap( -1 )
335         self.cross_auto.SetFont( wx.Font( 10, 74, 93, 92, False, "Arial
                Narrow" ) )
336         self.cross_auto.SetForegroundColour((0, 0, 128))
337
338         CorrelationSizer.Add( self.cross_auto, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
339
```

```
340         self.m_staticText15 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"Number of samples to be integred:", wx.DefaultPosition,
                wx.DefaultSize, 0 )
341         self.m_staticText15.Wrap( -1 )
342         self.m_staticText15.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial"
                ) )
343
344         CorrelationSizer.Add( self.m_staticText15, 0, wx.ALIGN_LEFT|wx.ALL,
                5 )
345
346         NSizer = wx.BoxSizer( wx.HORIZONTAL )
347
348         self.m_staticText19 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"N = ", wx.DefaultPosition, wx.DefaultSize, 0 )
349         self.m_staticText19.Wrap( -1 )
350         NSizer.Add( self.m_staticText19, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
351
352         self.N = wx.TextCtrl( self.CorrelationTab, wx.ID_ANY, N_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
353         NSizer.Add( self.N, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
354
355
356         CorrelationSizer.Add( NSizer, 1, wx.ALIGN_LEFT, 5 )
357
358         self.m_staticText16 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"Initial sample at which the correlation integral is computed:
                ", wx.DefaultPosition, wx.DefaultSize, 0 )
359         self.m_staticText16.Wrap( -1 )
360         self.m_staticText16.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial"
                ) )
361
362         CorrelationSizer.Add( self.m_staticText16, 0, wx.ALIGN_LEFT|wx.ALL,
                5 )
363
364         N0Sizer = wx.BoxSizer( wx.HORIZONTAL )
365
366         self.m_staticText20 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"n0 = ", wx.DefaultPosition, wx.DefaultSize, 0 )
367         self.m_staticText20.Wrap( -1 )
368         N0Sizer.Add( self.m_staticText20, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
369
370         self.n0 = wx.TextCtrl( self.CorrelationTab, wx.ID_ANY, n0_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
371         N0Sizer.Add( self.n0, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
372
373
374         CorrelationSizer.Add( N0Sizer, 1, wx.ALIGN_LEFT, 5 )
375
376         self.m_staticText17 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"Number of desired waveforms: ", wx.DefaultPosition,
                wx.DefaultSize, 0 )
```

```
377         self.m_staticText17.Wrap( -1 )
378         self.m_staticText17.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial"
                ) )
379
380         CorrelationSizer.Add( self.m_staticText17, 0, wx.ALIGN_LEFT|wx.ALL,
                5 )
381
382         WSizer = wx.BoxSizer( wx.HORIZONTAL )
383
384         self.m_staticText21 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"w = ", wx.DefaultPosition, wx.DefaultSize, 0 )
385         self.m_staticText21.Wrap( -1 )
386         WSizer.Add( self.m_staticText21, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
387
388         self.w = wx.TextCtrl( self.CorrelationTab, wx.ID_ANY, w_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
389         WSizer.Add( self.w, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
390
391
392         CorrelationSizer.Add( WSizer, 1, wx.ALIGN_LEFT, 5 )
393
394         self.m_staticText22 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"Correlation window: ", wx.DefaultPosition, wx.DefaultSize, 0 )
395         self.m_staticText22.Wrap( -1 )
396         self.m_staticText22.SetFont( wx.Font( 9, 74, 90, 92, False, "Arial"
                ) )
397
398         CorrelationSizer.Add( self.m_staticText22, 0, wx.ALIGN_LEFT|wx.ALL,
                5 )
399
400         MminstepSizer = wx.BoxSizer( wx.HORIZONTAL )
401
402         self.m_staticText23 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"mmin = ", wx.DefaultPosition, wx.DefaultSize, 0 )
403         self.m_staticText23.Wrap( -1 )
404         MminstepSizer.Add( self.m_staticText23, 0, wx.ALIGN_CENTER|wx.ALL, 5
                )
405
406         self.mmin = wx.TextCtrl( self.CorrelationTab, wx.ID_ANY, mmin_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
407         MminstepSizer.Add( self.mmin, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
408
409         self.m_staticText24 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"mstep = ", wx.DefaultPosition, wx.DefaultSize, 0 )
410         self.m_staticText24.Wrap( -1 )
411         MminstepSizer.Add( self.m_staticText24, 0, wx.ALIGN_CENTER|wx.ALL, 5
                )
412
413         self.mstep = wx.TextCtrl( self.CorrelationTab, wx.ID_ANY,
                mstep_value, wx.DefaultPosition, wx.DefaultSize, 0 )
414         MminstepSizer.Add( self.mstep, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
```

```
415
416
417         CorrelationSizer.Add( MminstepSizer, 1, wx.ALIGN_LEFT, 5 )
418
419         MmaxSizer = wx.BoxSizer( wx.HORIZONTAL )
420
421         self.m_staticText26 = wx.StaticText( self.CorrelationTab, wx.ID_ANY,
                u"mmax = ", wx.DefaultPosition, wx.DefaultSize, 0 )
422         self.m_staticText26.Wrap( -1 )
423         MmaxSizer.Add( self.m_staticText26, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
424
425         self.mmax = wx.TextCtrl( self.CorrelationTab, wx.ID_ANY, mmax_value,
                wx.DefaultPosition, wx.DefaultSize, 0 )
426         MmaxSizer.Add( self.mmax, 0, wx.ALIGN_CENTER|wx.ALL, 5 )
427
428
429         CorrelationSizer.Add( MmaxSizer, 1, wx.ALIGN_LEFT, 5 )
430
431
432         self.CorrelationTab.SetSizer( CorrelationSizer )
433         self.CorrelationTab.Layout()
434         CorrelationSizer.Fit( self.CorrelationTab )
435         ################################################################################
436
437         ##### ADD TABS TO THE INPUTS NOTEBOOK
                ################################################
438         self.InputsNb.AddPage( self.FileTab, u"Select file/s", False )
439         self.InputsNb.AddPage( self.CFDTab, u"Calibration, Filtering &&
                Doppler", False )
440         self.InputsNb.AddPage( self.CorrelationTab, u"Correlation", False )
441         ################################################################################
442         ################################################################################
443         ################################################################################
444
445         ##### CREATE OUTPUTS NOTEBOOK
                ########################################################
446         self.OutputsNb = wx.Notebook( self, wx.ID_ANY, wx.DefaultPosition,
                wx.DefaultSize, 0 )
447         self.OutputsNb.SetBackgroundColour( wx.Colour( 204, 229, 255 ) )
448         ################################################################################
449
450         ##### CREATE OUTPUTS TAB
                ############################################################
451         self.OutputsTab = wx.ScrolledWindow( self.OutputsNb, wx.ID_ANY,
                wx.DefaultPosition, wx.DefaultSize, wx.HSCROLL|wx.VSCROLL )
452         self.OutputsTab.SetScrollRate( 5, 5 )
453         OutputsSizer = wx.BoxSizer( wx.VERTICAL )
454
455         self.m_staticText1 = wx.StaticText( self.OutputsTab, wx.ID_ANY,
                u"Select output type", wx.DefaultPosition, wx.DefaultSize, 0 )
456         self.m_staticText1.Wrap( -1 )
```

```
457         self.m_staticText1.SetFont( wx.Font( 10, 74, 90, 92, False, "Arial"
                ) )
458
459         OutputsSizer.Add( self.m_staticText1, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
460
461         self.opt_plot = wx.CheckBox( self.OutputsTab, wx.ID_ANY, u"One plot
                for each configuration", wx.DefaultPosition, wx.DefaultSize, 0 )
462         self.opt_plot.SetValue(opt_plot_value)
463         OutputsSizer.Add( self.opt_plot, 0, wx.ALL, 5 )
464
465         self.opt_csv = wx.CheckBox( self.OutputsTab, wx.ID_ANY, u"One .csv
                file for each configuration", wx.DefaultPosition,
                wx.DefaultSize, 0 )
466         self.opt_csv.SetValue(opt_csv_value)
467         OutputsSizer.Add( self.opt_csv, 0, wx.ALL, 5 )
468
469         self.opt_waveform = wx.CheckBox( self.OutputsTab, wx.ID_ANY, u"One
                plot waveform cluster", wx.DefaultPosition, wx.DefaultSize, 0 )
470         self.opt_waveform.SetValue(opt_waveform_value)
471         OutputsSizer.Add( self.opt_waveform, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
472
473         self.opt_doppler = wx.CheckBox( self.OutputsTab, wx.ID_ANY, u"One
                plot Delay - Doppler map", wx.DefaultPosition, wx.DefaultSize, 0
                )
474         self.opt_doppler.SetValue(opt_doppler_value)
475         OutputsSizer.Add( self.opt_doppler, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
476
477         self.reset = wx.Button( self.OutputsTab, wx.ID_ANY, u"Reset input
                values", wx.DefaultPosition, wx.DefaultSize, 0 )
478         OutputsSizer.Add( self.reset, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
479
480         self.run = wx.Button( self.OutputsTab, wx.ID_ANY, u"Run",
                wx.DefaultPosition, wx.DefaultSize, 0 )
481         OutputsSizer.Add( self.run, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
482
483         self.no_errors = wx.StaticText( self.OutputsTab, wx.ID_ANY, "The Run
                button hasn't been clicked yet", wx.DefaultPosition,
                wx.DefaultSize, 0 )
484         self.no_errors.Wrap( -1 )
485         self.no_errors.SetFont( wx.Font( 9, 74, 93, 92, False, "Arial
                Narrow" ) )
486         self.no_errors.SetForegroundColour((49,127,67))
487
488         self.errors = wx.StaticText( self.OutputsTab, wx.ID_ANY, "",
                wx.DefaultPosition, wx.DefaultSize, 0 )
489         self.errors.Wrap( -1 )
490         self.errors.SetFont( wx.Font( 9, 74, 93, 92, False, "Arial Narrow" )
                )
491         self.errors.SetForegroundColour((255,0,0))
492
493
```

```
494         OutputsSizer.Add( self.no_errors, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
495         OutputsSizer.Add( self.errors, 0, wx.ALIGN_LEFT|wx.ALL, 5 )
496 
497 
498         self.OutputsTab.SetSizer( OutputsSizer )
499         self.OutputsTab.Layout()
500         OutputsSizer.Fit( self.OutputsTab )
501         ########################################################################################
502 
503         ##### ADD TABS TO THE OUTPUTS NOTEBOOK
                ############################################
504         self.OutputsNb.AddPage( self.OutputsTab, u"Outputs", False )
505         ########################################################################################
506 
507         ##### ADD NOTEBOOKS TO THE MAINSIZER OF THE FRAME
                ###################################
508         MainSizer.Add( self.InputsNb, 3, wx.EXPAND |wx.ALL, 5 )
509         MainSizer.Add( self.OutputsNb, 2, wx.EXPAND |wx.ALL, 5 )
510         ########################################################################################
511 
512         self.SetSizer( MainSizer )
513         self.Layout()
514 
515         ##### EVENTS
                ##########################################################################
516 
517         # Browse and select files
518         self.Bind(wx.EVT_BUTTON, self.BrowseClick, self.browse)
519 
520         self.Bind(wx.EVT_BUTTON, self.RunClick, self.run)
521 
522         self.Bind(wx.EVT_BUTTON, self.ResetClick, self.reset)
523 
524         self.Bind(wx.EVT_BUTTON, self.SelectFilterParam, self.filterbtn)
525 
526         self.Bind(wx.EVT_CHOICE, self.ChoiceCorrelation, self.x_choice )
527         self.Bind(wx.EVT_CHOICE, self.ChoiceCorrelation, self.y_choice )
528 
529     ##### Functions of the MainFrame Class
            ################################################
530     ########################################################################################
531 
532     # Check if a file or directory exists
533     def check_file(self, fil):
534         import os.path
535         if os.path.exists(fil):
536             return True
537         else:
538             return False
539 
540     # Check if some input parameters are numbers
```

```
541     def is_number(self, s):
542         try:
543             float(s)
544             return True
545         except ValueError:
546             pass
547 
548         try:
549             import unicodedata
550             unicodedata.numeric(s)
551             return True
552         except (TypeError, ValueError):
553             return False
554 
555     def check_parameters(self, param):
556         import unicodedata
557         param_str = ["fxmin", "fxstep", "fxmax", "N", "n0", "w", "mmin",
                "mstep", "mmax"]
558         param_mix = [0,0,0,0,0,0,0,0,0]
559 
560         # Check if the parameters are numbers
561         k = 0
562         cont = 0
563         err_num = ""
564         for i in param:
565             if self.is_number(i) == False:
566                 err_num = err_num + param_str[k] + ", "
567                 param_mix[k] = i
568                 cont = cont + 1
569             else:
570                 param_mix[k] = float(i)
571 
572             k = k + 1
573 
574         if cont == 1:
575             err_num = err_num[:-2] + " is not a number.\n"
576         elif cont > 1:
577             err_num = err_num[:-2] + " are not numbers.\n"
578 
579         # Check if the parameters are well introduced
580         err_well = ""
581 
582         # fxmin <= fxmax
583         if type(param_mix[0]) == float and type(param_mix[2]) == float:
584             if param_mix[0] > param_mix[2]:
585                 err_well = err_well + param_str[0] + " should be lesser or
                        equal than " + param_str[2] + ".\n"
586 
587         # fxstep > 0
588         if type(param_mix[1]) == float:
589             if param_mix[1] <= 0:
```

```
590                 err_well = err_well + param_str[1] + " should be positive.\n"
591             elif type(param_mix[0]) == float and type(param_mix[2]) == float:
592                 if param_mix[1] > (param_mix[2] - param_mix[0]) and
                        param_mix[0] <= param_mix[2]:
593                     err_well = err_well + param_str[1] + " should be at most
                            (fxmax-fxmin).\n"
594
595         # N > 0
596         if type(param_mix[3]) == float and param_mix[3] <= 0:
597                 err_well = err_well + param_str[3] + " should be positive.\n"
598
599         # w > 0
600         if type(param_mix[5]) == float and param_mix[5] <= 0:
601             err_well = err_well + param_str[5] + " should be positive.\n"
602
603         # n0 >= 0
604         if type(param_mix[4]) == float and param_mix[4] < 0:
605             err_well = err_well + param_str[4] + " should be positive or
                    zero.\n"
606
607         # mmin <= mmax
608         if type(param_mix[6]) == float and type(param_mix[8]) == float:
609             if param_mix[6] > param_mix[8]:
610                 err_well = err_well + param_str[6] + " should be lesser or
                        equal than " + param_str[8] + ".\n"
611
612         # mstep > 0
613         if type(param_mix[7]) == float:
614             if param_mix[7] <= 0:
615                 err_well = err_well + param_str[7] + " should be positive.\n"
616             elif type(param_mix[6]) == float and type(param_mix[6]) == float:
617                 if param_mix[7] > (param_mix[8] - param_mix[6]) and
                        param_mix[6] <= param_mix[8]:
618                     err_well = err_well + param_str[7] + " should be at most
                            (mmax - mmin).\n"
619
620
621         err = err_num + err_well
622
623         return err
624
625     # Change the text label of the correlation
626     def ChoiceCorrelation(self, evt):
627         x = self.x_choice.GetStringSelection()
628         y = self.y_choice.GetStringSelection()
629         if x==y:
630             self.cross_auto.SetLabel("AUTO - Correlation will be performed")
631             self.cross_auto.SetForegroundColour((128, 0, 0))
632         else:
633             self.cross_auto.SetLabel("CROSS - Correlation will be performed")
634             self.cross_auto.SetForegroundColour((0, 0, 128))
```

```
635
636     # Close the main frame
637     def OnTimeToClose(self, evt):
638         print "SoOp-GUI Closed!"
639         self.Close()
640
641     # Browse and select the data files
642     def BrowseClick(self, evt):
643         dlg = wx.FileDialog(
644             self, message="Choose a file",
645             defaultFile="",
646             wildcard=wildcard,
647             style=wx.OPEN | wx.MULTIPLE | wx.CHANGE_DIR
648             )
649         if dlg.ShowModal() == wx.ID_OK:
650             paths = dlg.GetPaths()
651             self.files.Clear()
652             for path in paths:
653                 self.files.AppendText(path+"\n")
654
655         files_selected_list = paths
656         dlg.Destroy()
657
658     # Browse and select the data files
659     def SelectFilterParam(self, evt):
660         dlg = wx.FileDialog(
661             self, message="Choose a file",
662             defaultFile="",
663             wildcard=wildcard,
664             style=wx.OPEN | wx.MULTIPLE | wx.CHANGE_DIR
665             )
666         if dlg.ShowModal() == wx.ID_OK:
667             paths = dlg.GetPaths()
668             self.filter_file.Clear()
669             for path in paths:
670                 self.filter_file.AppendText(path)
671
672         files_selected_list = paths
673         dlg.Destroy()
674
675     # Reset all the parameters entered
676     def ResetClick(self, evt):
677         default_param = [files_value, L1button_value, L5button_value,
            cal_on_value, cal_off_value, hx_on_value, hx_off_value,
            hy_on_value, hy_off_value,
678                         fxmin_value, fxstep_value, fxmax_value,
                            x_choice_value, y_choice_value, N_value,
                            n0_value, w_value,
679                         mmin_value, mstep_value, mmax_value, opt_plot_value,
                            opt_csv_value, opt_waveform_value,
                            opt_doppler_value, filter_file_value]
```

```
680
681         controls = [self.files, self.L1button, self.L5button, self.cal_on,
                self.cal_off, self.hx_on, self.hx_off, self.hy_on, self.hy_off,
682                     self.fxmin, self.fxstep, self.fxmax, self.x_choice,
                            self.y_choice, self.N, self.n0, self.w,
683                     self.mmin, self.mstep, self.mmax, self.opt_plot,
                            self.opt_csv, self.opt_waveform, self.opt_doppler,
                            self.filter_file]
684         k = 0
685         for i in controls:
686
687             if type(i) == wx._controls.Choice:
688                 i.SetSelection(default_param[k])
689             else:
690                 i.SetValue(default_param[k])
691             k = k + 1
692
693         self.cross_auto.SetLabel("CROSS - Correlation will be performed")
694         self.cross_auto.SetForegroundColour((0, 0, 128))
695
696     # Detection of errors and starts the process of passing the parameters
            to the executable
697     def RunClick(self, evt):
698
699         open_plots = False
700
701         # Getting parameters values
702         files_val = str(self.files.GetValue().replace("\n"," "))
703         L1button_val = str(self.L1button.GetValue())
704
705         cal_on_val = str(self.cal_on.GetValue())
706         hx_on_val = str(self.hx_on.GetValue())
707         hy_on_val = str(self.hy_on.GetValue())
708
709         fxmin_val = str(self.fxmin.GetValue())
710         fxstep_val = str(self.fxstep.GetValue())
711         fxmax_val = str(self.fxmax.GetValue())
712
713         # 0 --> UP, 1 --> DW
714         x_choice_val = str(self.x_choice.GetSelection())
715         y_choice_val = str(self.y_choice.GetSelection())
716
717         N_val = str(self.N.GetValue())
718         n0_val = str(self.n0.GetValue())
719         w_val = str(self.w.GetValue())
720         mmin_val = str(self.mmin.GetValue())
721         mstep_val = str(self.mstep.GetValue())
722         mmax_val = str(self.mmax.GetValue())
723
724         opt_plot_val = str(self.opt_plot.GetValue())
725         opt_csv_val = str(self.opt_csv.GetValue())
```

```
726         opt_waveform_val = str(self.opt_waveform.GetValue())
727         opt_doppler_val = str(self.opt_doppler.GetValue())
728         filter_val = str(self.filter_file.GetValue())
729 
730         """
731         compiler = "g++ -I/home/gauss/vmoreno/lib/include/waveform_pylib_inc
                -L/home/gauss/vmoreno/lib/lib main_example.cpp -o main
                -lwaveform"
732         os.system(compiler)
733         """
734 
735         # Inputs that should be introduced in numbers
736         self.control_numbers = [fxmin_val, fxstep_val,
737                                 fxmax_val, N_val,
738                                 n0_val, w_val, mmin_val,
739                                 mstep_val, mmax_val]
740         err_text = self.check_parameters(self.control_numbers)
741 
742         def_files = "No selected files"
743         def_filter = ""
744 
745         self.no_errors.SetLabel("")
746 
747         # No filter file is necessary when hx and hy are off
748         if hx_on_val == "False" and hy_on_val == "False":
749             filter_val = "False"
750 
751 
752         if err_text == "" and files_val != def_files and filter_val !=
                def_filter:
753             self.no_errors.SetLabel("Input parameters OK.")
754             self.errors.SetLabel("")
755 
756         err_file = "No data file is selected."
757         err_filter = "No filter file is selected."
758 
759         if err_text != "" and files_val == def_files and filter_val ==
                def_files:
760             self.errors.SetLabel(err_text +"\n" + err_file + "\n" +
                    err_filter)
761 
762         elif err_text != "" and files_val == def_files:
763             self.errors.SetLabel(err_text +"\n" + err_file)
764 
765         elif err_text != "" and filter_val == def_filter:
766             self.errors.SetLabel(err_text +"\n" + err_filter)
767 
768         elif files_val == def_files and filter_val == def_filter:
769             self.errors.SetLabel(err_file +"\n" + err_filter)
770 
771         elif err_text != "":
```

```
772             self.errors.SetLabel(err_text)
773
774         elif files_val == def_files:
775             self.errors.SetLabel(err_file)
776
777         elif filter_val == def_filter:
778             self.errors.SetLabel(err_filter)
779
780         else:
781             open_plots = self.check_file("./main")
782             if open_plots == True:
783                 self.no_errors.SetLabel("Results computed.")
784                 executer = ("./main "+ files_val + " " + cal_on_val + " "
785                 + hx_on_val + " " + hy_on_val + " "
786                 + fxmin_val + " " + fxstep_val + " "
787                 + fxmax_val + " " + x_choice_val + " "
788                 + y_choice_val + " " + N_val + " "
789                 + n0_val + " " + w_val + " "
790                 + mmin_val + " " + mstep_val + " "
791                 + mmax_val + " " + opt_plot_val + " " + opt_csv_val + " "
792                 + opt_waveform_val + " " + opt_doppler_val + " "
                        +filter_val+" "+L1button_val)
793
794                 os.system(executer)
795                 print "Works correctly"
796                 #app2 = PlotsApp() MIRAR POR QUE DA SEGMENTATION FAULT
797                 #app2.MainLoop()
798             else:
799                 self.errors.SetLabel("Executable not found.")
800
801
802 #### App for observe the output results
    ##################################################
803 ##########################################################################################
804 class PlotsApp(wx.App):
805     def __init__(self, redirect=False, filename=None):
806         wx.App.__init__(self, redirect, filename)
807         self.SecondFrame = wx.Frame(None, title='PLOT RESULTS')
808         self.panel = wx.Panel(self.SecondFrame)
809         self.PhotoMaxSize = 640
810         self.createWidgets()
811         self.SecondFrame.Show()
812
813     def createWidgets(self):
814         img = wx.EmptyImage(640,360)
815         self.imageCtrl = wx.StaticBitmap(self.panel, wx.ID_ANY,
816                                          wx.BitmapFromImage(img))
817
818         instructLbl = wx.StaticText(self.panel, label = 'Browse for a PNG
            image')
819         self.photoTxt = wx.TextCtrl(self.panel, size=(400,-1))
```

```
820         browseBtn = wx.Button(self.panel, label='Browse')
821         browseBtn.Bind(wx.EVT_BUTTON, self.Browse)
822 
823         self.MainSizer = wx.BoxSizer(wx.VERTICAL)
824         self.sizer = wx.BoxSizer(wx.HORIZONTAL)
825 
826         self.MainSizer.Add(wx.StaticLine(self.panel, wx.ID_ANY),
827                            0, wx.ALL|wx.EXPAND, 5)
828         self.MainSizer.Add(instructLbl, 0, wx.ALL, 5)
829         self.MainSizer.Add(self.imageCtrl, 0, wx.ALL, 5)
830         self.sizer.Add(self.photoTxt, 0, wx.ALL, 5)
831         self.sizer.Add(browseBtn, 0, wx.ALL, 5)
832         self.MainSizer.Add(self.sizer, 0, wx.ALL, 5)
833         self.panel.SetSizer(self.MainSizer)
834         self.MainSizer.Fit(self.SecondFrame)
835 
836         self.panel.Layout()
837         self.panel.SetBackgroundColour( wx.Colour( 194, 219, 245 ) )
838 
839     def Browse(self, event):
840         wildcard = "PNG files (*.png)|*.png"
841         dialog = wx.FileDialog(None, "Choose a file",
842                                wildcard=wildcard,
843                                style=wx.OPEN)
844         if dialog.ShowModal() == wx.ID_OK:
845             self.photoTxt.SetValue(dialog.GetPath())
846         dialog.Destroy()
847         self.Watch()
848 
849     def Watch(self):
850         filepath = self.photoTxt.GetValue()
851         img = wx.Image(filepath, wx.BITMAP_TYPE_ANY)
852         W = img.GetWidth()
853         H = img.GetHeight()
854         if W > H:
855             NewW = self.PhotoMaxSize
856             NewH = self.PhotoMaxSize * H / W
857         else:
858             NewH = self.PhotoMaxSize
859             NewW = self.PhotoMaxSize * W / H
860         img = img.Scale(NewW,NewH)
861 
862         self.imageCtrl.SetBitmap(wx.BitmapFromImage(img))
863         self.panel.Refresh()
864 
865 if __name__ == "__main__":
866     app = wx.App()
867     frame1 = MainFrame()
868     frame1.Center()
869     frame1.Show()
870     app.MainLoop()
```

D.3. Matlab scripts

D.3.1. Main

```
clear all; close all; format longG; clc;

% Ground Station (GS)
lat_GS = 41.500436; lat_GS_rad = lat_GS*pi/180;
long_GS = 2.110422; long_GS_rad = long_GS*pi/180;
h_GS = 139; % AMSL [m]
h_GS_agl = 16; % [m]

% Geostationary Satellite: ASTRA 1M & 1KR [n2yo.com]
lat_1m = 0;
long_1m = 19.2;
lat_1kr = -0.02;
long_1kr = 19.18;

var = 0;
%var = input('ASTRA 1M --> 0 or ASTRA 1KR --> 1 \n');
if (var == 0)
    lat_sat_rad = lat_1m*pi/180;
    long_sat_rad = long_1m*pi/180;
end
if (var == 1)
    lat_sat_rad = lat_1kr*pi/180;
    long_sat_rad = long_1kr*pi/180;
end

% Orbital altitude respect to the Earth surface [m]
Ro = 35786e3;
% Earth radius in the Ground Station [m]
[ Rt ] = EarthRadius( lat_GS, h_GS ); % Rt is the Earth radius + height
% above sea level

% Distance Satellite - GS [m]
[x_GS,y_GS,z_GS] = lla2ecef(lat_GS_rad, long_GS_rad, h_GS);
coord_GS = [x_GS,y_GS,z_GS];
[x_sat,y_sat,z_sat] = lla2ecef(lat_sat_rad, long_sat_rad, Ro);
dGS_sat = PitagorasTheorem(x_GS,y_GS,z_GS, x_sat,y_sat,z_sat);

% Distance GS - Scatterer & Distance Satellite - Scatterer

input_data = load('input_data.txt');
lat_scat = input_data(:,2); long_scat = input_data(:,3); h_scat =
    input_data(:,4);
lat_scat_rad = lat_scat*pi/180; long_scat_rad = long_scat*pi/180;

for i=2:length(input_data)

```

```
    [x_scat,y_scat,z_scat] = lla2ecef(lat_scat_rad(i),
        long_scat_rad(i),h_scat(i));
    coord_scat(i,:) = [x_scat,y_scat,z_scat];
    dGS_scat(i,1) = PitagorasTheorem(x_GS,y_GS,z_GS, x_scat,y_scat,z_scat);

    dSat_scat(i,1) = PitagorasTheorem(x_sat,y_sat,z_sat,
        x_scat,y_scat,z_scat);

end

% Delay distance and Samples

fm = 80e6; % [Hz]
c = 3e8;

d1 = dGS_sat;
for i=2:length(input_data)

    d2(i,1) = dSat_scat(i) + dGS_scat(i);
    Dd(i,1) = d2(i) - d1;

    samples(i,1) = Dd(i)/(1/fm*c);

end

% Pointing angles of the UP Antenna
%latitude and longitude with their respective signs MIRAR DE CAMBIAR LA
    FUNCION GRADOS-RADIANES
[ az_UP, el_UP ] = PointingAngles( lat_GS, long_GS, long_sat_rad*180/pi,
    'n', 'e', Rt, Ro );

% Pointing angles of the DW Antenna

for i=2:length(input_data)
    [ az_DW, el_DW] = PointingDW( coord_GS, coord_scat(i,:));
    anglesDW(i,1) = az_DW; anglesDW(i,2) = el_DW;
end
% [ az_DW, el_DW ] = PointingDW( coord_GS, [0,0,0]);
% anglesDW(i+1,1) = az_DW; anglesDW(i+1,2) = el_DW;
%
% dGS_scat(i+1,1) = PitagorasTheorem(x_GS,y_GS,z_GS, 0, 0, 0);
%
% [ az_DW, el_DW ] = PointingDW( coord_GS, [0,0,0]);
% anglesDW(i+2,1) = az_DW; anglesDW(i+2,2) = el_DW;
%
% dGS_scat(i+2,1) = PitagorasTheorem(x_GS,y_GS,z_GS, 0, 0, 0);

%Dd is the difference distance
M1 = [input_data, dGS_scat, Dd, samples, anglesDW ];
xlswrite('output_data.xlsx',M1, 1, 'B2:J30');
```

```
92 M2 = {'Name', 'Latitude [deg]', 'Longitude [deg]', 'Altitude [m]', 'GS -
       Scatterer distance [m]', 'Difference distance [m]', 'Samples', 'DW
       Azimuth [deg]', 'DW Elevation [deg]'};
93 xlswrite('output_data.xlsx',M2, 1, 'B1:J1');
```

D.3.2. Functions

```

1
2 function [x,y,z]=lla2ecef(lat,lon,alt)
3
4 % WGS84 ellipsoid constants:
5 a = 6378137;
6 e = 8.1819190842622e-2;
7
8 % intermediate calculation
9 % (prime vertical radius of curvature)
10 N = a ./ sqrt(1 - e^2 .* sin(lat).^2);
11
12 % results:
13 x = (N+alt) .* cos(lat) .* cos(lon);
14 y = (N+alt) .* cos(lat) .* sin(lon);
15 z = ((1-e^2) .* N + alt) .* sin(lat);
16
17 end
```

```
1 function [ A, E ] = PointingAngles( lat_GS, long_GS, long_sat, ns, we, Rt,
       Ro )
2
3 % A: Azimuth of the pointing antenna [deg]
4 % E: Elevation of the pointing antenna [deg]
5
6 % lat_GS : latitude of the Ground Station [deg]
7 % long_GS : longitude of the Ground Station [deg]
8 % long_sat : relative satellite longitude [deg]
9 % Rt : Earth radius in the Ground Station [m]
10 % Ro : Orbital altitude respect to the Earth surface // Distance between
       both locations [m]
11 % ns : location (North = 'n' or South = 's')
12 % we : location (East = 'e' or West = 'w')
13
14 lat_GS = deg2rad(lat_GS); long_GS = deg2rad(long_GS); long_sat =
       deg2rad(long_sat);
15
16 % G = Difference between satellite orbital position and earth station
       antenna.
17
18 % 1) If the satellite orbital location is in east (E),
19 % then G = Antenna longitude - Satellite orbital position.
```

```
% 2) If the satellite orbital location is in West (W),
% then G = Satellite orbital position - Antenna longitude

if (we == 'e') == 1
    G = long_GS - long_sat;
    a = 0;
end

if (we == 'w') == 1
    G = long_sat - long_GS;
    a = 1;
end

% The azimuth (A) is the angle between the ground station meridian and
% the local vertical plane that contains the satellite pointing direction.

if (ns == 'n') == 1
    if a == 0
        A = 180 + rad2deg(atan(tan(G)/sin(lat_GS)));
    end
    if a == 1
        A = 180 - rad2deg(atan(tan(G)/sin(lat_GS)));
    end
end

if (ns == 's') == 1
    if a == 0
        A = 360 - rad2deg(atan(tan(G)/sin(lat_GS)));
    end
    if a == 1
        A = rad2deg(atan(tan(G)/sin(lat_GS)));
    end
end

E = rad2deg(atan((cos(G)*cos(lat_GS)-Rt/(Ro +
    Rt))/(sqrt(1-cos(lat_GS)^2*cos(G)^2))));

end
```

```
function [ azimuth, elevation] = PointingDW( coord1, coord2 )

% Latitude and Longitude in radians
% coord1 and coord2 are the coordinates (x,y,z) of two locations in the ECEF
% reference system in meters

% azimuth and elevation are returned in degrees [deg]

%% after

[lat1, long1, alt1] = LLA(coord1(1), coord1(2), coord1(3));

```

```
lat1 = deg2rad(lat1);
long1 = deg2rad(long1);
%
% disp(lat1*180/pi);
% disp(long1*180/pi);

Su = [sin(lat1)*cos(long1), sin(lat1)*sin(long1), -cos(lat1)];
Eu = [-sin(long1), cos(long1), 0];
Zu = [cos(lat1)*cos(long1), cos(lat1)*sin(long1), sin(lat1)];

Nu = -Su;

rho = (coord2-coord1);

S = dot(Su, rho);
E = dot(Eu, rho);
Z = dot(Zu, rho);

elevation = asin(Z/norm(rho))*180/pi;
azimuth = atan2(E, -S)*180/pi;


 rhou = (coord2-coord1)/norm(coord2-coord1);

elevation2 = 90 - rad2deg(acos(dot(Zu, rhou)));

% rhoproj = [dot(rhou,Su), dot(rhou, Eu), dot(rhou, Zu)];
% rhoproju = rhoproj/norm(rhoproj);
% azimuth = rad2deg(acos(dot(Nu, rhoproju)));




end
```